

contents

- 04. Why should the government consider FLOSS
- 07. Coming out of your shell
- 12. Why I use KDE
- 14. Fedora 13 Released
- 17. ސަލާމް ދެކުމާ ފަދަ ގޮތް ގަތުމަށް ފަސޭހަ ވާ ގޮތް ދައްކާ ގޮތް
- 18. Understanding Linux : Part I

contact us

Maldives Open Source Society Website
<http://moss.org.mv>

Sending Your Voice
magazine@moss.org.mv

Maldives Linux User Group/MOSS Mailing List
<http://groups.google.com/group/mlugmv>

Launch Pad Dhivehi Translators Mailing List
<http://groups.google.com/group/divtranslators>

MOSS Technical Support Forum
<http://groups.google.com/group/moss-technical>

WE ARE ACCEPTING DONATIONS!

MOSS is now accepting donations. Why now, after one year? Well, we felt we should have a funding goal before accepting donations, and somehow it didn't feel right until we'd toiled for a year, just to proof our passion.

Our funding goals are simple - as MOSS grows we have more demanding needs that costs money. In addition, now that MOSS has successfully run for a year and there are impressive projects and activities being planned out, we feel it's the right time to think about wider promotion.

So, if you've benefitted from MOSS, believe in what we stand for and would like to give something back (other than actively involving in our activities, which is still probably the best way to contribute), we'd appreciate any donation, large or small.

Just deposit your donation to our account:

Bank of Maldives
 7701-179480-001
 Maldives Open Source Society

Thank you in advance for all your help and support as we work together to advocate the benefits and beauty of FLOSS in the Maldives. Rest assured that we are committed to providing and advocating FLOSS with all that we have got.

GIVE TO GROW
Together



MOSS: Your Monthly FLOSS Magazine is freely available at moss.org.mv.

For any questions or comments about MOSS magazine, we can be contacted at :

magazine@moss.org.mv

The articles contained in this magazine are released under the Creative Commons Attribution-Share Alike 3.0 Unported license. This means you can adapt, copy, distribute and transmit the articles but only under the following conditions: You must attribute the work to the original author in some way (at least a name, email or URL) and to this magazine by it's name ('MOSS') and the URL www.moss.org.mv (but not attribute the article(s) in any way that suggest that we endorse you or your use of the work). If you alter, transform, or build upon this work, you must distribute the resulting work under the same, similar or a compatible license.



COVER June 2010, ISSUE #06

Maldives Open Source Society

STEERING COMMITTEE

President
Ismail FAIZ (PH)

Vice President
Yusuf Abdulla SHUNAN

Secretary
ROBERT Boettcher

Public Relations
Ibrahim SOBAH

Program Coordinator
Mohamed MALIK

Treasurer
HUSSAIN Sharaah

Advisory Board
INASH Zubair – Founding Member
Mohamed VISHAH – Founding Member

Magazine Editors
Ahmed SHUJAAU Mohamed
cosmicflu
SIMON Shareef
SOUL

THE READERS' VOICE

Readers' opinions regarding our magazine and/or previous articles.

MOSS is all about Software Freedom. If you think you could be part of MOSS or you know someone who could, please let us know, we want our magazine to be interactive. We want to progress with you. The common use of free software is not as far off as you think, so let's open our minds to the world and the future. Let's make it happen!

If you have any good ideas or constructive opinions regarding MOSS magazine's contents, this is the place to express yourself.

Please email us at magazine@moos.org.mv



Why should the Government consider FLOSS...

➔ [Mohamed MALIK > mohamedmalik.com](#)

Looking at the history of FLOSS, way back 15 years ago, we can say FLOSS was born out of capable hobbyist's seeking freedom, wanting to share and help one another in the arena of software development. However, today it has grown larger than just a hobbyist's toy. FLOSS has grown and is growing at an exponential rate in its development and adaptation that need a corporate touch. With a quick look at some key players like Canonical, Red Hat, Novel, IBM and others, who have embraced FLOSS as a business with success. How they make a profit may be a surprise to some. In some cases they charge a low priced annual subscription for constant update and support services. For example, Linux Red Hat Enterprise Linux (RHEL), is maintained and fully supported by Open Source Software. They charge for the support starting from \$80 per year on a subscription basis. This does not mean users have to pay in-order to make use of the functionality of RHEL; CentOS contains binaries (software) compiled from the same RHEL source code stripped of the RHEL logos and branding, for the benefits of those who do not wish to pay a subscription and for those who have the knowledge to self maintain a system without support dependencies.

Even then there are free Server versions like Ubuntu Server which is supported by both by the community showing the true FLOSS spirit at the same time with the option of commercial support by Canonical.

In short, Microsoft is a working platform at a relatively huge cost while FLOSS is a reliable platform that is flexible and economical. Having said that, most significant advantages of FLOSS are that it comes at zero licensing hassles, unlimited customization and the potential benefits of using a more versatile operating system. It is not only about the software, I am referring to more about the solution, both short and long term.

Throughout the world from Europe to Asia to Latin America, politicians and bureaucrats have considered FLOSS for variety of reasons. In Japan, where they take reliability and availability of their mission-critical applications in their system as a higher priority are deploying FLOSS, to prevent vendor locks and the benefits of global standards.

In the European Union, government technocrats are examining FLOSS as a way to smooth the integration of conflicting, parochial software and communications standards. And in Latin America, where concerns over skyrocketing

software fees and Microsoft market hegemony have triggered a spate of reactionary legislative bills, the debate surrounding open source software carries both nationalist and populist overtones.

For the most part, however, government agencies are exploring open source options as a way to enhance software flexibility, lower costs and, to improve system reliability as more government services aim to reach the community online. FLOSS has flourished in those places where users view software not as a political game but as a pragmatic tool. Given the political environment, the best thing to say about any software program is that it should get the job done and stop at that.

Now comes the question of do Maldives have the expertise to handle FLOSS platforms such as Linux? The answer is, we already have proven Linux back-end implementations, notably in Dhiraagu, Wataniya, Island Aviation, Focus Infocom and others. All these systems were setup and are currently maintained by local expertise. That tackles the most challenging tasks, and for the workstations and desktop the learning curve and adaptability is easier than one might think. So in this economically challenging times, Government and Businesses considering FLOSS is a must.

Then comes the question what will happen to our country with the implementations of the copyright law. Many individuals and even the government still does not understand the meaning of copyright. It means duplicating or copying or cracking any software, movie or using something that is copyrighted under international law will be an offense, punishable by law.

In countries like United States the federal law punishes people with up to \$250,000 or

In countries like United States the federal law punishes people with up to \$250,000 or up to five years of prison.

up to five years of prison imprisonment. Here in the majority of the Maldivian society, when we consider any home or small cooperate business and most of the government offices, they run counterfeit software products. Meaning that if the copyright law is implemented in the Maldives home users, businesses and the government will not have any other choice other than buying the software from the respective vendors. This is when FLOSS comes into play. Proprietary software is very expensive to buy. Consider this example. A web developer working in a government office needs or uses the following software packages. He would most definitely need Adobe Master Suite Collection, which costs \$2,400. He would also need an operating system that supports that software mostly Microsoft windows 7. It costs yet another \$250. That means that only the operating system and the Adobe creative suits cost the government \$2,650, which is RF34,052 for just one single person. And what about all the other employees who are working in the office, they also need to run software on there computers which would help them to do there daily tasks. Which means that for every computer that the government has, the government has to buy the software for that particular computer.

The real question now would be, will it be possible for the government to spend millions of dollars just to buy the softwares that they need. At the moment our society has been hit by an economic crisis, the prices of things has gone so high that the public are complaining about it. Can the government sacrifice developmental plans and budgets allocated to other areas to buy software? Or will they take a loan from the world bank to run all the systems in the government? If any sorts of this happens I believe that the results will be catastrophic.

Many countries have been motivated to switch to FLOSS alternative, the French police is one good example. Consider France, it is a country which has resources, man power and money, but why would they drop proprietary software and go for the alternative, the answer is simple, FLOSS software has any advantages from closed coded proprietary software which I will be taking a look later. Did the French police made the transformation overnight? They did not do it overnight. First they changed the office software to OpenOffice.org the free office and creativity suite. Then later they replaced the entire operating system platform to Ubuntu. According to the French police by the transformation they have saved more than 70% of the money that required to run, maintain and purchase proprietary software. They claim that they have saved 50 million Euro by making the transformation. For government and businesses saving money means a lot, they could spend that money to do something which is more useful and beneficial. The only benefit of embracing FLOSS software is not all about saving money. It is also about what it can do, something that the closed coded software does not allow us to do. FLOSS software means that the source code is available. By using the source code of the software anyone with programming knowledge will be able to modify it to his or her needs. They can share the software, make as many copies as they like to, clearly the advantages are simple and logical to understand.

As the copyright law comes to effect, the government needs to device a plan, transforming to a new change. Can we handle this change? Whatever we have to do, we need to carefully plan and implement. One way to do it is to take a step by step approach in transforming all the computers to FLOSS software. Or just go for an over-night change, a complete change. However we want to change, there is enough reason and benefits for the Government to consider FLOSS.

I believe the current ruling government came in power in the vision for a change, for a good change. What I am highlighting is a similar change, a good change. If a 30 year old regime can be changed for the better, I believe

Coming Out of Your Shell

Linux Shell Commands (Part 1)

By *Ahmed Sunil*
(ahmedsunil@aol.com, linux-zero.blogspot.com)

Learning to navigate the terminal is the first step in mastering the power of Bash. After all, you have to know where you are going and how to get there.

Press alt+f2, this will open the run applications box, type gnome-terminal and press enter.

path

Using Paths in Linux

The concept of paths needs a little bit of explanation. Every user, when they log in, has a default path. Find your default path with the following command:

```
$ echo $PATH
```

The output should look something like this:

```
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/nill/bin
```

The \$PATH is a list of directories separated by colons. If a command is typed without a path, then all the directories in the default path are checked, in order, for the file associated with the command. In the above example, if there is a command named guess in both the /usr/local/bin and the /home/nill/bin directories, the one in the /usr/local/bin is executed. If you'd rather use the guess command in the /home/nill/bin directory, then you'll need to use the full path, i.e., type the /home/nill/bin/guess command.

find, locate, and grep

The find, locate, and grep commands are powerful tools for searching for files. This section discusses only their most basic uses because it would be easy to devote a full chapter to using these commands. If you want a full discussion of the commands, check their man pages once you learn to use the man command later in this chapter.

While all three commands are used for searching, their purposes differ: find is used to search for files by any number of criteria, including name or date of creation, while grep is used to search the contents of files.

find

find

The command can be used to search for files by name, date of Creation or modification, owner (usually the user who created the file), size of the file, and even type of the file. This section describes the most frequent use: searching for files by name. The basic structure of the find command is

```
$ find starting-directory parameters actions
```

The starting directory specifies where to begin searching. For instance, specifying /home means only subdirectories of /home will be searched (in other words, only the user's home directories will be searched), while specifying / means everything will be searched.

The parameters are where you specify the criteria by which to search. In this case, you use the command switch -name filename to specify the file you are searching for. The actions section indicates what action to take on found files. Generally, you will want to use the -print action, which indicates that the full name and path of the file should be displayed. Without this, the find command will perform the search indicated but will not display the results, which defeats the purpose. Putting this together, if you want to search for all files named foo on your system, you could use the command

```
$ find / -name foo -print
```

In this case, the results look like this:

```
$ find / -name foo -print
/tmp/foo
/home/armand/foo
/home/tdanesh/foo
```

It is also possible to search for partial filenames. For instance, if you know that the file you are looking for begins with fo, then you can use the expression fo* to indicate all files beginning with fo and ending with any combination:

```
$ find / -name 'fo*' -print
/tmp/foo
/var/lib/texmf/fonts
/usr/bin/font2c
/usr/bin/mh/folders
/usr/bin/mh/folder
/usr/bin/mh/forw
```



```

/usr/bin/formail
/usr/bin/fontexport
/usr/bin/fontimport
/usr/bin/fold
etc.

```

Notice the use of the single quotation marks around `fo*`. When you use the `*` character, it is important to place the single quotes around the entire expression.

Otherwise, `find` will give you an error:

```

$ find / -name fo* -print -mount
find: paths must precede expression
Usage: find [path...] [expression]

```

If the results being produced by the `find` command are too numerous to fit in one screen, you can use piping and the `more` command just like you did earlier with the `ls -l` command:

```

$ find / -name 'fo*' -print | more

```

locate

`locate`

If the `find` command takes too long, you may be able to use the alternative, the `locate` command. This command searches a database of files on your system, created nightly. It works slightly differently from the `find` command, since it returns every file and directory with your search string in its name. For example, the `locate xauth` command would give you these results:

```

$ locate xauth
/home/nill/.xauth
/home/nill/.xauth/refcount
/home/nill/.xauth/refcount/root
/home/nill/.xauth/refcount/root/testlinux
/lib/security/pam_xauth.so
/usr/X11R6/bin/mkxauth
/usr/X11R6/bin/xauth
/usr/X11R6/man/man1/mkxauth.1x.gz
/usr/X11R6/man/man1/xauth.1x.gz
/usr/share/doc/pam-0.72/txts/README.pam_xauth
/usr/share/man/man8/pam_xauth.8.gz

```

Observe how this command gives you the full path to all files and directories that include the text string “xauth,” including the `/home/nill/.xauth` directory and the `/usr/X11R6/bin/mkxauth` and `/usr/X11R6/bin/xauth` commands.

This works a lot more quickly than the corresponding `find` command. The drawback is that `locate` works from a database file that is updated generally only once every 24 hours; therefore, the results may not reflect the current location or even the existence of a recently moved or created file.

grep

`grep`

Where `find` searches for a file by its name, type, or date, and `locate` searches through a file database, `grep` is used to look inside the contents of one or more files in an attempt to find the occurrence of a specific pattern of text inside the files.

Consider an example. You know that you created a text file that contains the word “radio” and stored it in your home directory. However, you have forgotten the name of the file and want to quickly check which files contain “radio.” This is where `grep` comes in handy.

Assuming you are in your home directory, the following command searches for the word “radio” in each file in your home directory and produces results as follows:

```
$ grep radio *
ab.txt:This is a test of searching for the word radio.
pop.txt:On another radio station, he found that
```

Notice how the `grep` command returns one line for each occurrence of the word “radio” in a file. The name of the file is shown followed by a colon, which is followed by the complete text of the line where the word appeared.

In general, the pattern for the `grep` command is

```
$ grep text-pattern file-list
```

The text pattern can be a simple word or phrase or a more complicated regular expression. (The use of regular expressions—a powerful method for searching for text patterns—with `grep` can be found in the `grep` man page.) The file list can take any form allowed by the shell. Consult Chapter 16 for a complete discussion of the types of expressions that constitute a file list.

Generally, though, you want to either check the contents of a single file, which takes the form

```
$ grep text-pattern file-name
```

or check the contents of all files in a directory using the command

```
$ grep text-pattern *
```

where the `*` is an expression indicating that all files in the current directory should be searched.

In its simplest form, the text pattern is a single word or part of a word containing no spaces. If you want to search for a phrase, such as “is a test,” you need to enclose the text pattern in quotation marks, as in the following example:

`$ grep “is a test” * ab.txt`: This is a test of searching for the word radio. Just as it is sometimes useful to pipe the results of a command through the more or less commands, the same is true for `grep`. Consider the situation where you want a listing of all files in the current directory with the modification date of May 12. You can find this information by piping `ls -l` through a `grep` command:

```
$ ls -l | grep “May 12”
-rw-r--r--  1 root    root      19197 May 12 21:17 rfbprotoheader.pdf
-rw-r--r--  1 root    root     110778 May 12 21:20 rfprotoA.zip
-rw-r--r--  1 root    root     17692 May 12 23:03 svnc-0.1.tar.gz
-rw-r--r--  1 root    root     25222 May 12 19:58 vnc-3.3.1_javasrc.tgz
drwxr-xr-x  2 root    root      1024 May 12 21:49 vncjava
```

These are some commands that are useful in Linux. If I get feedbacks, I will write second part for this tutorial. The information on this article is not fully my ideas. But most are. And I took examples from some references.

Why I use KDE?

Mohamed Malik > mohamedmalik.com

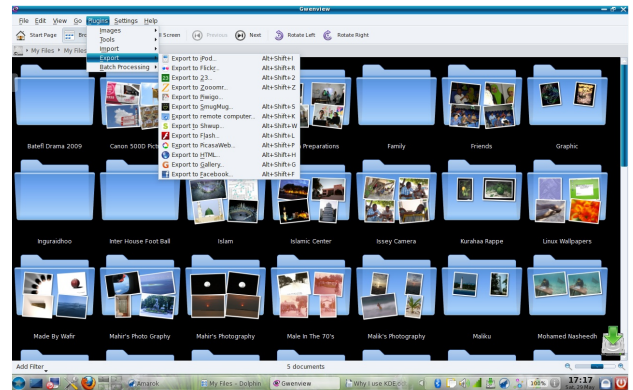
Many people, especially the hard working people in MOSS ask me the question why I prefer KDE over GNOME. Firstly, in point of view it depends on personal preference. If the desktop environment can do what I need to do in my daily life and activities that means that I do not have any problems with it. Here are some of the reasons why I prefer KDE instead of GNOME.

Many claim that KDE4 is not ready for production work yet. I totally disagree to this because those who say this have only used KDE4 under Ubuntu's official derivative Kubuntu. No offence to anybody who uses Kubuntu, but I personally feel that the KDE4 delivered with Kubuntu is very unstable. The KDE4 in Kubuntu is just the basic KDE4 with no tweaks or customizations' made by the Kubuntu Team. However recent rumors point that the Kubuntu Team is working very hard to improve their KDE4 desktop. Think about this, the GNOME environment that is supplied with Ubuntu has been tweaked very well; therefore it works like a charm in Ubuntu. No other distro provides a better integration with GNOME and the system than in Ubuntu, the one that comes close is Fedora, since Fedora also uses GNOME as its default. If you have used a distro like Opensuse or Mandriva, the KDE that comes with these distro's are very stable and suitable for production machines. This also happens due to the fact that KDE is the default desktop environment for these distributions; therefore they work very hard to improve it.

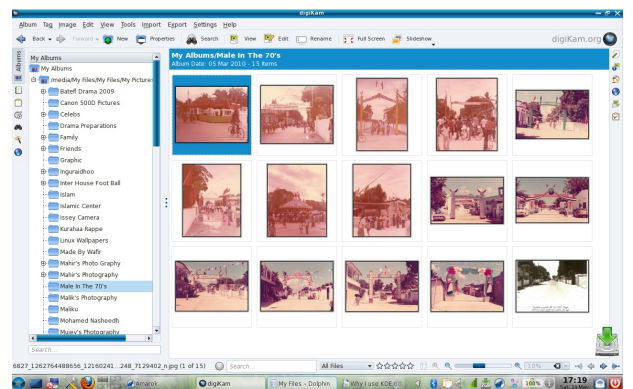
Ok! With the initial release of KDE4.0 in January of 2008, it was a disappointment I wouldn't argue with that. Even Linus Torvalds said the same and it made him switch to GNOME. However with the release of KDE4.2, 4.3 and 4.4 KDE is back in full force I would say.

As someone who maintains an online community news website and a personal blog, KDE gives me all the features I need. GNOME is built for simplicity however KDE is made for raw power and with the end user in mind. The default image viewer in GNOME, Eye of the GNOME has basic features that a typical user would want in an image viewer. However KDE's image viewer Gwenview is packed with more features. It can edit, resize pictures, upload pictures to almost any online imaging service platform. This is the reason why I prefer it. As someone who updates websites frequently I need to resize, crop and edit my pictures before uploading it. Gwenview gives me all I need. However in GNOME I would need to open

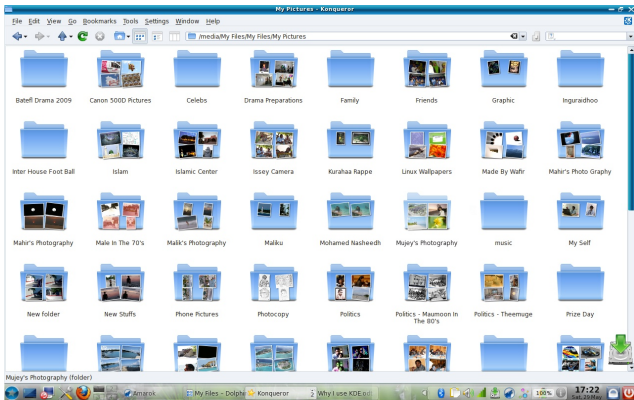
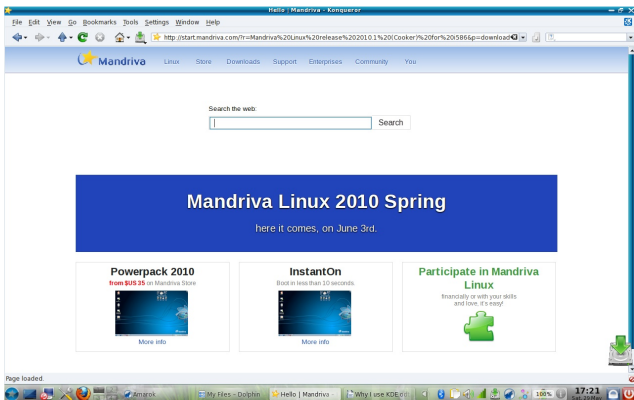
GIMP if I'm resizing pictures. Given the option of uploading pictures to flickr, I could do that as well without installing any other software or without opening my web browser. It also allows the option of browsing all my pictures without the need for opening the file manager.



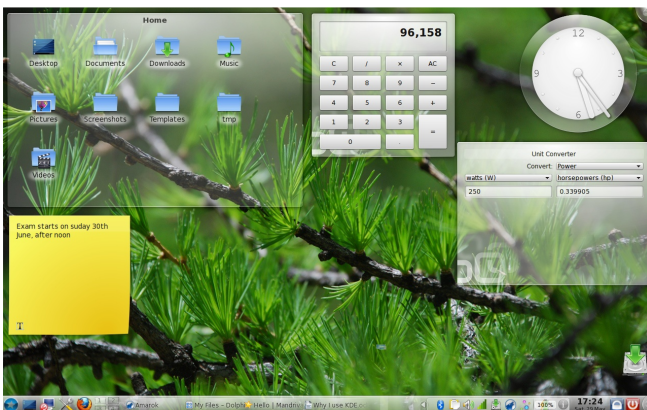
KDE's image management program Digikam is a very useful program, since I'm someone who has over 60GB's of pictures in my collection. I need a program that can organize and manage all the pictures in my collection. It allows me to manage my photographs like a professional. Editing, framing, resizing, 16bit color management is available. It is the one and only application that is available for Linux that has 16bit color management. As an image management program GNOME offers F-spot which doesn't give me all the features I need.



The other reason is Konqueror. I can Konquer my desktop with it. The question is how? Konqueror is the all in one web browser, file manager and document viewer. Meaning that I could do three things at once while I use the program, manage, organize my files, browse the World Wide Web, and use it to view all my documents without having to open any other software.



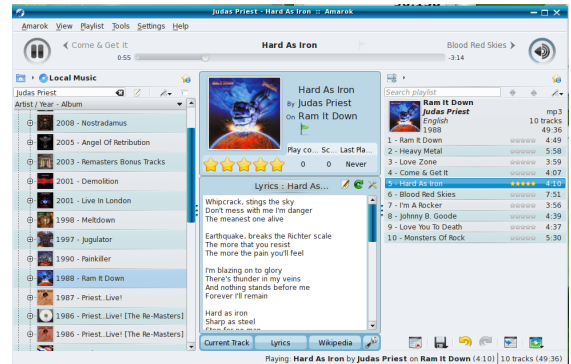
Since the introduction of plasma desktop and plasmoids in KDE, it has made my desktop much more functional and better. The weather applet shows me the weather. I can use the calculator to do all my calculations. In addition to that it also provides me with the option of a unit converter plasmoid, as a physics teacher this is one of the useful plasmoids that are available in KDE.



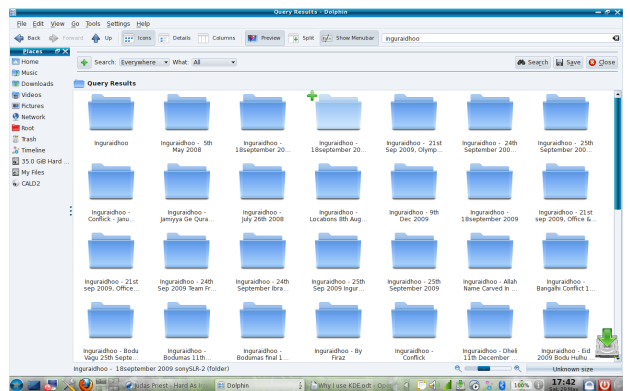
Music lovers and people who have a large collection of songs need an audio player that can fulfil his/her needs. Amarok just gives me what I need? The reason why I prefer Amarok is the features that come along with it. While playing music, I can read the artist info which is provided with an applet in the centre of Amarok. It also gives me the option to read the lyrics while I'm listening to my music. Similar artist and related

artist fields are also very useful. It also gives me the option to rate my music collection, create unlimited amount of playlists, mass tagging of songs, and more importantly it looks great too!

The integration of Nepomuk search and indexing



tool is another great feature which has been added to KDE recently. With the files indexed in Nepomuk users can search their files by file names or tags or with the extension of the file format. It lets users to find files quickly and efficiently.



These are some of the reasons why I prefer KDE over GNOME. However I believe that GNOME is a great desktop environment as well. Especially for users who want simplicity and just getting things done. But for an end user like me I would say that KDE would provide you with a much better and more powerful desktop environment and a much better experience with it. Well that is just the beauty of Open source and Linux systems. We have a wide variety of choices available. For a new user I would say give everything a try, whether it's GNOME or KDE, even XFCE and decide what suits you best. We don't need to be stuck like closed coded software users who are bound to use one thing whether they like it or not.

Fedora 13 Released

An Overview

Author: cosmicflu

A rough 6 months after Fedora 12, the gurus at Fedora Project has released the much awaited Fedora 13, aka "Goddard", named after the rocket scientist Robert Hutchings Goddard (and hence its marketing slogan "rock it" for this release). Fedora 13 promises to offer an extensive range of leading edge features, not to mention its collection of current software, improved user-friendly design and technological advances that will probably end up in many other distributions as well.

PC World has called it a "clean and fast" Linux desktop distribution. I would label it as a distribution which brings joy and excitement of experiencing new innovations while not sacrificing stability.

Fedora 13 has come a long way and has reached a certain level of maturity, for the fact that the scale has been weighed more towards stabilizing the current packages, albeit keeping in mind its relatively "heavy" on both sides. Time and time again, the Fedora community has shown the world that they will continue to develop and intergrate the latest FLOSS solutions in its distributions.

Borrowing from the official website and others, some of the major changes seen in Fedora 13 compared to its predissesor are highlighted below.



Underlying methodology that Fedora adopts, i.e having well written and tested third party software, and being utilized as a testing ground for the super stability of Red Hat Enterprise Linux provides, made it possible for the community to accept it as one of the most widely used Linux distributions.

One Liners

- Automatic print driver installation
- Automatic language pack installation
- Redesigned user account tool
- Better color management to calibrate monitors and scanners
- Enhanced DisplayPort support in Nvidia and ATI graphics cards
- Experimental 3D support for NVIDIA graphics cards
- A new way of installing Fedora over the Net
- NFS updates and enhanced support for the experimental Btrfs
- Zarafa Open Source Edition
- Firefox 3.6.3, Thunderbird 3.0.4, GIMP 2.6.8, OpenOffice 3.2.0 etc
- Support for entire Java EE 6 spec in NetBeans 6.8

Latest version of GNOME 2.3 (released in March) has been integrated, and discussions are underway to shift to GNOME 3 in Fedora 14 (as there will not be any additions to the GNOME series 2 development anymore). It has been a while since the GNOME Shell (being one of the main components of next-gen of GNOME) was integrated into Fedora, hence its a safe bet when saying that Fedora 13 offers the maximum stability (with respect to GNOME) any Linux distribution on the scene can currently offer. As for KDE, version 4.4.2 is used (and it uses Qt 4.6). Currently KDE 4.4.3 is also available as a test update and will pass on as a regular update soon enough.

Although most packages (such as OpenOffice and GIMP) has not been included in the installation media due to space limitations, they are easily installable directly from the software repositories that come with Fedora 13. One particularly compelling addition noteworthy of mentioning that has been included in this version of Fedora is the Zafra Open Source edition suite. It is a groupware suite that provides integration with existing Linux mail servers, is said to be 100% compatible with Exchange environments and even has its Ajax web-interface customized to look like that of Outlook. Look out, no excuses now in going for a replacement of Exchange! Here's what Fedora had to say about Zafra:

"Fedora 13 now makes available a complete Open Source groupware suite that can be used as a drop-in Exchange replacement for Web-based mail, calendaring, collaboration and tasks. Features include IMAP/POP and iCal/CalDAV capabilities, native mobile phone support, the ability to integrate with existing Linux mail servers, a full set of programming interfaces, and a comfortable look and feel using modern Ajax technologies. Fedora encourages contributors to package and maintain 100% free, unencumbered software that is useful to our users. This package is provided in Fedora thanks to the work of long-time Fedora volunteer Robert Scheck, who packaged it for inclusion in the distribution."

Btrfs (an experimental file system for Linux which is being heavily developed, which although might contain bugs now, is probably on its way in becoming the "Next Generation File System for Linux") is still supported and has been further enhanced, particularly its snapshot feature. Now with Btrfs, it enables Fedora users to switch back to a prior snapshot of the system. Say for example if it is found that after updating the system, the updates turned out to be unstable, now you have the choice of reverting the system to a state prior to the updates being applied.



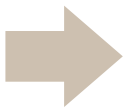
Fedora 13 continues to lead in the Linux file system world; it uses ext4 as the default file system now. Fedora deserves that pride for the simple reason that Fedora was the first Linux distribution to include it, and now it is the first to offer ext4 by default!



The kernel in Fedora 13 is based on Linux version 2.6.33.3. Speaking of whom, **the Author of the Linux Kernel, the genius Linus Torvalds himself, prefers Fedora as his distribution of choice**, and continues to use it from 2008 onwards. It can be noted that Fedora changes the kernel version, sometimes several times, in a single development branch; Fedora 12 was originally released with kernel version 2.6.31 and was recently (before Fedora 13) was updated to version 2.6.32, but has not been updated to include the 2.6.33 which was included in Fedora 13 when it was released. An update to the recently released kernel version 2.6.34 is already being developed in the update package repositories!

However, updating a new kernel is always performed while keeping the option of booting from the already existing kernel version prior to the update. This ensures that users still access an otherwise deadlocked system resulting from a kernel update. A Fedora kernel update brings numerous goodies along with it; new hardware support and stable drivers. Users of other mainstream distributions such as Ubuntu have to wait even months until a main release with a recent kernel supporting hardware components, unless they switch to a developers version of their distribution.

An exciting new way of installing Fedora has been introduced with this version, the boot.fedoraproject.org (BFO). It is somewhat similar to the pxeboot environment. The concept uses a lightweight boot image to bootstrap a host system, which then contacts remote servers on the internet for boot information. This means that the installation is initiated from the remote packages, hence eliminating the need to update the boot image even if Fedora releases change. **Fedora suggests that the BFO concept might replace huge DVD/CD ISO downloads in the future thus saving on unnecessary bandwidth and installation time, and revolutionize the way we install Linux distributions.**



Different flavours or "spins" of Fedora continues to be available even with this release. These ISO images of live media (software range customized for specific user groups) include the well known variants of components GNOME, KDE, LXDE, Xfce along with new spins being introduced with Fedora 13; "Design Suite" spin for graphics designers, "Security" spin aimed at security auditing, forensics, system testing and recovery, and various others where one focuses on low powered mobile devices as netbooks ("Moblin" spin) and a "Sugar on a Stick" spin with the award winning Sugar Learning Environment integrated in it and designed to be carried on a USB thumb drive.

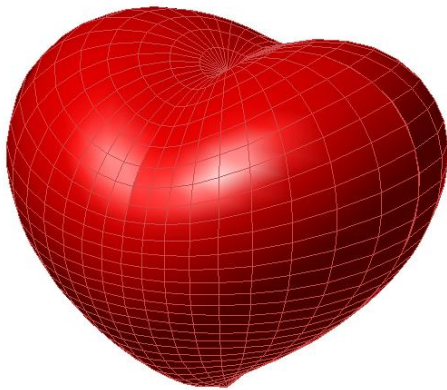
Get yourself one of these spins and start experiencing Fedora now!



Understanding Linux : Part 01

The Heart of Linux

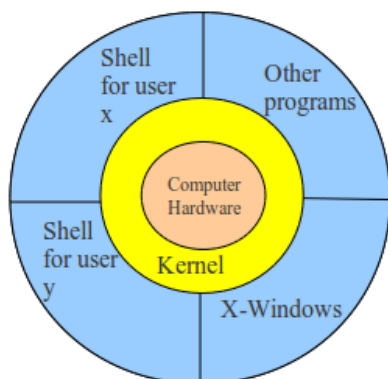
Yusuf Abdulla Shunan | shunan@gmail.com



This series of articles (Understanding Linux) will give you a quick inside tour of Linux, answering questions like what is Linux, Shell, Kernel and how your valuable files are handled in Linux? What actually happens when you type a command? In all demonstrations, I will be using Ubuntu Linux; Ubuntu is arguably the most well known desktop Linux distribution at street level. For many new Linux users Ubuntu is Linux and Linux is Ubuntu, however there are many different flavors of Linux. Ubuntu is relatively easy to install, setup and manage, as by design it is made so, hence Ubuntu is attractive to the first time users. Ubuntu has a strict six-month release cycle, meaning a new release comes every six months. On most machines Ubuntu installation is relatively automatic and the underlying power of the command line is nicely hidden from new users, only available to those who crave for adventure and dare. However even in Ubuntu users do have the opportunity to explore beyond the Graphical User Interface (GUI). So once a new user is brave enough to explore these powers it is time for the real hacking fun and it is also the time to explore other choices. In the world of Linux choices are ample and a users freedom is only limited by imagination. If a user needs to change something, they are free to do it, even to the point to just screw things up. So if a user find Ubuntu restrictive in any means they have the choice of other distros like Fedora, Mandriva, Kubuntu, CentOS, Gentoo, Mepis and PCLinuxOS, to name a few!

Today, the introduction to Linux for a general user usually goes through an introduction of Ubuntu to other distributions, and this series of articles (Understanding Linux) are only for the brave, who wants to explore the ins and outs of Linux. These articles are designed for those brave souls who want to know how things work, so let's begin with the Heart of Linux, the Kernel.

The Kernel is actually the main program that control the computer's resources, distributing them to different users and to different tasks. However, the Kernel do not directly interact with the users but it uses an interactive program for each user who logs on, when logged on a terminal, this is always a shell. The shell from there on acts as an interface acting in between the user and the system. It takes up the role of as a command interpreter, the shell takes users inputs and get them up for execution. However it is the Kernel that handle the constant demands of the shells of different users and decides exactly when and how each requested program should be executed. Thus, the kernel and your shell collaborate to provide



Why is there just one Kernel but several shells and programs? The reason for multiple shells and programs is that Linux can handle more than one user or task at a time. Usually, one computer “process” goes on at any given moment, but Linux can switch rapidly from one process (such as your shell) to another (such as `man`’s command), giving the illusion of multi-tasking. The different shells provide a way to separate one user or task from another, while the Kernel maintains coherent overall control.

When you log on, it is the Kernel that runs `init` and `getty` to check to see if you are an authorized user and have the correct password. The Kernel keeps track of all the various programs being run, allocating time to each, deciding when one stops and another starts. The Kernel assigns storage for your files. The Kernel runs the shell programs. The Kernel handles the transfer of information between the computer and terminals, tape drives and printers. In other words, the Kernel is the heart of a Linux system, which is the reason why it is the Kernel.

One of the most important Kernel duties is to run the time-sharing/dividing system. Linux is a multiuser, multitask, multi-threaded system. Multiuser means that more than one person (or other inorganic beings) can use the computer at the same time. Multitask means that even a single user can have the computer work on more than one task or program at the same time. Multi-threaded means one complex task can be divided into several tasks to be solved simultaneously using multiple CPUs. In Linux, each particular task or program that the computer is undertaking is called a process. For example, when you type the command `date` on a terminal, that’s one process. If you run it again, then that’s a new process.

Normally, a Linux system has several processes to handle at the same time. However, up until recent times the computer that we used had single track minds; they can only do one thing at a time. This is because they had a single CPU. One of the current

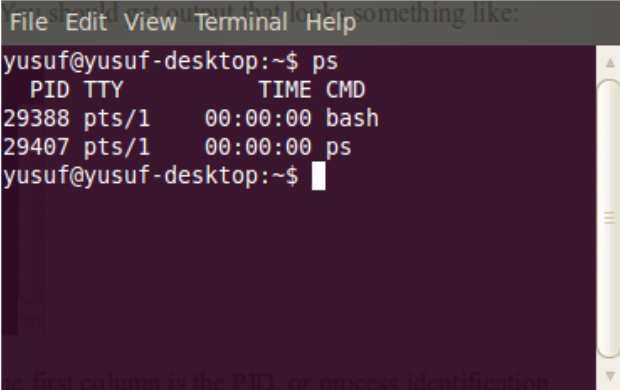
machines that can do several tasks simultaneously. As you know today CPU’s with 8 cores are common for the desktop. Linux is the dominant multi-CPU (super computers, render farms, etc.) operating system and the current Linux Kernel splits tasks up amongst the different cores as it is. In addition programs can be made to explicitly make use of multiple cores. Both Intel and AMD have opened up the routines for multi-threaded programming under open source licenses and with OpenMP and GOMP multi-threaded programming can be handled at the programming level (while distributed, clusters & grids can be managed using MPI and MOSIX2). However, multi-threaded programming is a difficult task and it is not viable for all programs, so for the sake of simplicity let us only concentrate on how Linux operates on a single CPU.

A single CPU Linux uses time-sharing to solve the problem of multiple demands upon a single track mind. Time sharing means the Kernel maintains a list of current tasks (or processes) and allocate a bit of time to one process, then to the next, and so on, sharing the available time sequentially among the waiting processes. Typically, the Kernel switches from one process to process so fast that users see it as that they have the undivided attention of the computer. But in a single CPU if the work load get too heavy, you may find yourself waiting even between hitting a key and seeing the corresponding letter appear on the screen.

At any given time on a Linux system, many processes are generally active. The one process that is actually running at a given moment is known to have run status. The ones that are waiting their turn usually have sleeping status (there are other statuses that I will not go into). If you want to see processes on a Linux system, you can use the `ps` (process status) command. Let’s see what that command show us about processes.

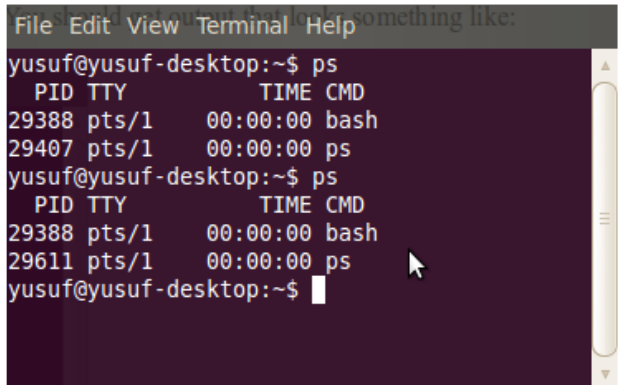
Just open up a terminal and type `ps` or you can login to a shell using your username and password and use any of the six shells supplied by the default Ubuntu Linux, which you can access using `Ctrl + Alt + F1` for `tty1` and `Ctrl+Alt+F2` for `tty2` and so on (note `F1/F2` are the function keys at the very top of your keyboard and to get back to X/GUI use `Ctrl+Alt+F7`).

When you type the `ps` command you should get an output that looks something like this...



```
File Edit View Terminal Help something like:
yusuf@yusuf-desktop:~$ ps
  PID TTY          TIME CMD
 29388 pts/1        00:00:00 bash
 29407 pts/1        00:00:00 ps
yusuf@yusuf-desktop:~$
```

The `ps` command makes a process status report. The first column is the PID, or process identification number. Each time a process is initiated, the Kernel assigns to it a unique PID. That is how the Kernel keep track of the different processes. In the example above the only two processes are the user shell itself (bash) and the `ps` command (29407). Each has its own PID and between the two, 19 processes were created (29407 - 29388 = 19). Note that for the Kernel the shell is just another process. If you type the same command again you will get a similar output as the second screenshot.



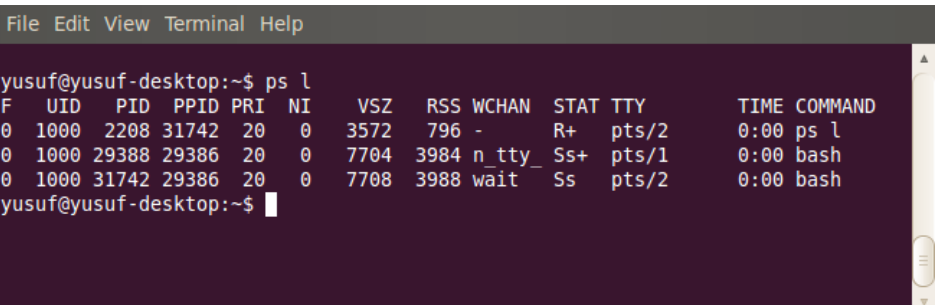
```
File Edit View Terminal Help something like:
yusuf@yusuf-desktop:~$ ps
  PID TTY          TIME CMD
 29388 pts/1        00:00:00 bash
 29407 pts/1        00:00:00 ps
yusuf@yusuf-desktop:~$ ps
  PID TTY          TIME CMD
 29388 pts/1        00:00:00 bash
 29611 pts/1        00:00:00 ps
yusuf@yusuf-desktop:~$
```

Note the bash shell has the same PID (29388) as we did not close the terminal so the process idles and when the command is issued the process continues. However the second `ps` command is a new process as the previous `ps` command expired after displaying the result, so the second `ps` command's PID has changed (29611). Every time you start a new task the shell will start a new process with a new PID to do the task.

The TTY column in the example indicates the terminal or other input device that you are using. The TIME column tells how much computer time the process has used. While the CMD or command column gives the name of the command corresponding to the process.

In geek terms the bash process in this example is called the parent, and the two `ps` processes created by the shell are children. Unlike the real world, here the parent will wait until the child dies.

So far we have just looked at your processes, but `ps` has options to give more detailed information. The option `l` (simple letter L) provides more information about each process and it is known as long format. While `-e` will show every process on the system using standard syntax and `aux` will show every process on the system using BSD syntax. The actual output will



```
File Edit View Terminal Help
yusuf@yusuf-desktop:~$ ps l
F  UID  PID  PPID  PRI  NI   VSZ  RSS  WCHAN  STAT  TTY          TIME COMMAND
0  1000  2208 31742  20   0   3572  796  -      R+    pts/2        0:00 ps l
0  1000  29388 29386  20   0   7704  3984 n_tty_ Ss+   pts/1        0:00 bash
0  1000  31742 29386  20   0   7708  3988 wait_  Ss    pts/2        0:00 bash
yusuf@yusuf-desktop:~$
```

This listing includes additional information, like Status (STAT) information. The only process running (R+) is, as you can see, the ps command itself, for it had to be running to produce the report. Everything else is practically sleeping (Ss) including my shell, which is waiting for the ps command to finish. The PPID is the Parent PID. Though this is a very brief list of the processes, I hope it showed the process-scheduling aspect of the Kernel duties.

Another important aspect of the Kernel is that it serves as an interface between the shell and the Linux commands on one hand and the system hardware on the other hand. When you type something at the keyboard, the Kernel collects your input and delivers it to the shell. When you use the cp command to copy a file, the Kernel finds space on the disk (if your system uses a disk for storage) for the new file and keeps track of the relevant information, such as the location and size of the file.

The Kernel is the part of Linux that directly communicates with the hardware. Process scheduling and the examples we just mentioned illustrate that. When Linux is ported (adapted) to a new kind of computer, it is the Kernel that has to be modified in order to make communication possible. But Kernel insulates the user from direct contact with the hardware. For example, we don't need to know what goes on inside the machine in order to see the Linux cp command. We can even write a new cp program without knowing about the innermost secrets of the hardware, for Linux lets us tap

In Linux, system calls are basic routines used in the Kernel and which you can use in a C program. When you make a system call, the portion of the Kernel with the relevant programming is run. If you use a system call to open a file, then the part of the Kernel that opens files is invoked in your behalf. Each system call accomplishes a specific basic task. One system call open a file, another lets you read a file, another closes a file, another launches a new process, and so on. The system calls are part of the Kernel and are used to do it's work.

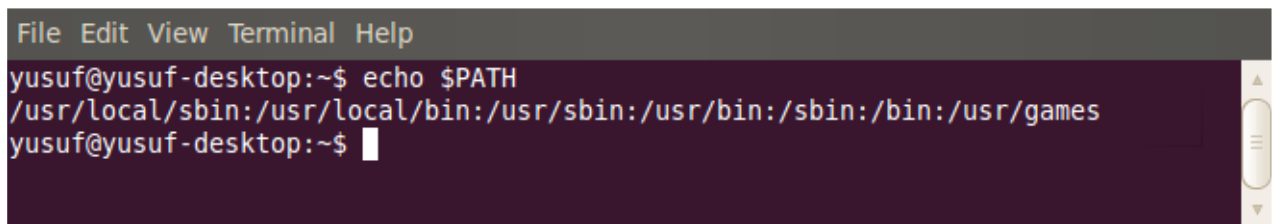
The system calls are also basic building blocks upon which Linux commands (like cp) are based. The programming for a command may use system calls directly by name, or indirectly, by using Linux library functions that in turn use system calls. The important point is that regardless of what computer we use, we can use the same system calls. That is, the hardware instructions for opening a file on a super computer like a Cray may be different from those for opening a file on a VAX/VMX, Mac or an IBM PC, but for all four (if we use Linux) we can use the open() system call to do the job. The system call approach helps make Linux easily portable. Reprogram the system calls for a new type of computer, and they will support the entire Linux superstructure.

Although the Kernel is really the head honcho, you normally deal with its emissary, the shell. It is the interface between you and the computer. To see how the shell works, let's

start by examining what happens when you type a command.

What happens when you type a command such as `date`? Try it! The easy answer is that the current time and date appear on the terminal screen, but our real interest is in what goes on behind the screen.

Let's begin at the very start (a reasonable



```
File Edit View Terminal Help
yusuf@yusuf-desktop:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
yusuf@yusuf-desktop:~$
```

choice). When you log in, the kernel starts up a program (the shell) that handles your interaction with the system. The shell sends a prompt symbol (typically a `$`) to the screen, then waits for input from you. When you type a command, (date for example) and hit the Enter key, the shell obtains your command, executes it if possible, and returns a prompt to you when it is done and ready for your next command. (The kernel does the actual fetching and returning of data at the request of the shell). Because the shell translates your typed commands to actions, it is termed a command interpreter (the Linux shell is also a programming language).

How does the shell go about its tasks of translating commands? There are a few built-in commands that are part of the shell, but that vast majority of commands are separate programs stored elsewhere in the system. When given, say, the command `date`, the shell searches for a file name `date`. If it finds such a file and if the file contains an executable program, the shell starts the program running and goes into a waiting mode. When the program finishes, the shell resumes.

In looking for the command file, the shell searches through a predefined list of directories given by the `PATH` shell variable. Typically this list includes your current working directory and system directories devoted to storing files of commands. The usual directories for this purpose are `/bin` and `/usr/bin`. To see what directories are in your search path. Just type: `echo $PATH`

Here the colon is used to separate directories, thus this example instructs the shell to look for a command first in `/usr/local/sbin` directory, then in the `/usr/local/bin` directory, next in the `/usr/sbin`, followed by `/sbin` and next `/bin` and last in the directory `/usr/games`. If it doesn't find the command in one of these directories, the shell reports back that the command was not found. Thus for the `date` command, for example, to be recognized, it must be stored in one of the directories in this list.

Most Linux commands themselves are compiled C programs, and the remaining few are shell scripts, which are files containing other Linux commands.

The Linux system includes a vast number of programs that you can use. We can view them as being arranged in a hierarchy of dependence. The most basic programs are the system calls, which form part of the kernel program. Typically, the programming of a call varies from one hardware system to another, but the name and the function of a system call is the same for all computers running Linux.

Next, there is a library of programs in the C programming language. These C library programs use the system calls to handle the interface between the programs and the hardware. Since different systems have the same calls, the library, in principle, is portable. That is, once you write a program based on library functions, you can transport it from one Linux-based machine to another.

Then there are the Linux commands. They represent the level at which most users deal with Linux. The commands are programs like `who` and `cat` and `ls` and so on. These commands, for the most part, are written in the C language and use system calls and/or C library programs.

Finally, some commands are shell scripts. That is, they are files that contain a list of other Linux commands. When the shell script command is run, what actually happens is that the list of commands in the file gets run.

So to conclude when you log in, the Kernel starts a shell process for you. When you type a command, the shell process uses the Kernel to start up a process for the corresponding command. Then the shell waits for this new child process to complete and die. Then the shell process resumes running. Each process is assigned its own PID. This lets the Kernel know which process is which.

The shell has many talents we have not yet discussed. There is one more point I would like to emphasize about the shell. The point is this: for each user a separate shell procedure is started up. Your shell doesn't pay attention to Mau's commands, and Mau's shell is unaware of your commands. The time-sharing system makes this possible. Last, shell, Kernel, system calls, commands, all these are programs with codes stored in files on a Linux system, so stay tuned for the part 2 of Understanding Linux, which will take a closer look at the Linux File System.

This Magazine was created using FLOSS

Graphic Design *GIMP*

<http://www.gimp.org/>

Layout Setting *Scribus*

<http://scribus.net/>

Type Setting *OpenOffice.org*

<http://www.openoffice.org/>

Operating System *Ubuntu 10.04 Lucid Lynx*

<http://www.ubuntu.com/>

Release License *Creative Commons*

<http://www.creativecommons.org/>

Creative Commons is a nonprofit corporation dedicated to making it easier for people to share and build upon the work of others, consistent with the rules of copyright.

Creative Commons provide free licenses and other legal tools to mark creative work with the freedom the creator wants it to carry, so others can share, remix, use commercially, or any combination thereof.