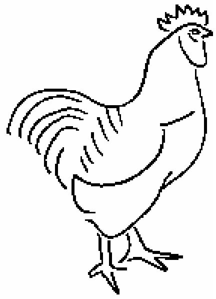
CLTC Project Information Sheet 0:

The GNU/Linux Operating System Explained



Developed by the ’Free Range’ *Community−Linux Training Centre Project* −
Version 1.0, August 2002.   http://www.fraw.org.uk/cltc/

This, rather wordy introduction to the Linux system is intended to provide background information on Linux, and what it is. It does not provide any practical guidance. Instead it conveys the theoretical concepts that will help you understand how the Linux operating system works.

In the beginning, there was Unix...

...and Unix was developed as a modular, networked, multi−tasking, multi−user operating system that was pretty much platform independent as it was coded in a standardised programming language − C.

’Multi−tasking’ meant that it could run many ’processes’ (programs) at the same time, and ’Multi−user’ meant that more than one person could use the operating system at the same time.

’Modular’ meant that the operating system was made up of many small building blocks − each building block carrying out a simple function − interacting together to produce the whole operating system.

’Networked’ essentially means that Unix is designed to operate across an environment of connected machines − acting as a ’server’ for a number of client machines. Mostly these client machines would be dumb terminals where the employees of a large corporation or the students at a university would use the functions of the computer. But a client could also be another Unix server located in another building, or another country.

Today, when the Internet is widely available, this sounds simple. But in the early 1970s, when Unix was being developed, it was a revolutionary ’common platform’ for small (by 1970s standards) ’minicomputers’. Previously computers had often used unique operating systems developed by the manufacturer of the system. Despite its power, from the point of view of the growing band of ’computer hackers’ who were just beginning to play with microcomputers, there were two problems with Unix. Firstly, Unix would not run on a system with the miniscule system resources of early microcomputers. Secondly, Unix belonged to someone − it wasn’t free for anyone to use and play with.

There were a number of attempts to re−create Unix for the IBM−compatible PC. In the early 1980s two companies, Santa Cruz Operation (SCO) and Microsoft collaborated to produce a PC−compatible version of Unix, called *Xenix*. It was proprietary, like Unix. But because PC’s didn’t have the power of their minicomputer cousins, it never really took off. In the mid−1980s, a project was started by Andrew Tanenbaum to produce a free Unix−like operating system, called *Minix*. But again, it never took off, because of the problems of developing Minix as a fully−functioning operating system (Tanenbaum began the development of Minix as part of an educational project).

The late 1980s saw an explosion in the use of the Internet at Universities and other educational establishments. There was also a growth in the use of microcomputers by amateurs and student hackers. People wanted the power of a Unix−like system, but at the time the only system that was developing for the IBM PC was Microsoft’s *MS−DOS* (Microsoft Disk Operating System) and later *Windows*. But the Internet allowed a new way of working, working collaboratively to develop a new projects. One computer enthusiast in particular was able to grasp this concept − *Linus Torvalds*.

GNU/Linux is born

Richard Stallman is the founder of the Free Software Foundation, and the initiator of the GNU Project (GNU − ’Gnu’s not Unix). His aim was to develop a free version of Unix. From the mid−1980s onwards he worked to develop free versions of the programming utilities people needed to develop computer systems. The heart of this process was the ’GNU Public License’ − a unique approach to licensing software that acknowledged the role of the creator of a program, but allowed others to copy and use the code for free.

By early 1991, Stallman had the various free utilities, but he lacked the ’kernel’ − the core of the operating system that runs the machine. At this time a 21−year old Finnish computer student, Linus Torvalds, bought an IBM PC. He used *MS−DOS* for a while, but shortly after obtained a copy of *Minix* − Tanenbaum’s free Unix project − and started playing with it. This was possible because Stallman's GNU−licensed software ran on the Minix OS.

After playing with Minix for a few months, Torvalds embarked on writing his own 'small' operating system. But rather than doing it all himself, he invited others to join in. This meant that rather than relying on his own resources and abilities, a project could be developed involving many people, of many different talents could between them wield the time and resources of a large computer corporation (the principle that still underpins many of the Linux−based projects underway today). Between mid−1991 and early 1994, the Linux OS developed from a minute kernel to a full Unix−like operating system. Linux was finally released as 'version 1.0' on March 13th 1994.

With the Linux kernel complete, the stage was now set for the development of free software projects. Stallman had paved the way by originating the GNU Public License, and assisting with the development of the utilities required to undertake system engineering. Torvalds originated the basics of a kernel, and worked with others to produce a working operating system in 1994. *GNU/Linux* (being a symbiotic relationship between Stallman's GNU project and Torvalds' kernel) was under power, and began replicating its way across the Internet − gathering more users, and spawning many new projects to improve the kernel, and develop new applications to use the Linux OS.

Today, Linus Torvalds still guides the development of the kernel in collaboration with many 'kernel hackers' across the globe. At the same time many projects have grown up around the development of the kernel that are equally significant − and all symbiotically related like the original complementary work of Stallman and Torvalds. For example, the *XFree86* Project works to develop the graphical functions of the Linux system. In turn, various groups use the graphical capabilities provided by *XFree86* to develop graphical user interfaces (GUIs) such as *Gnome* or *KDE*. Other developers then use the graphical environment of *Gnome* or *KDE* to develop their own applications to meet a perceived need for new programs or functionality amongst the Linux community.

Linux grows and develops 'distributions'

The first main point about Linux is that 'Linux' is not a complete operating system. It is just the kernel. An operating system requires a kernel, plus a large number of other programs and utilities to make it a viable system for the computer user. Therefore different groups have developed different 'distributions', each providing a slightly different set of applications to the others.

There are a growing number of distributions available. There are a few that are widely used, such as Red Hat, SuSE, Debian and Mandrake. Then there are many others that have evolved to suit specific needs − such as secure information servers. Some systems that appear to be one thing, such as a stand alone firewall, are actually a Linux distribution that works solely as a firewall and nothing else. The beauty of the Linux kernel is that because, like Unix, it is modular, you can swap and modify the blocks that make up the system to tailor its operation towards specific goals.

The choice of distribution is largely down to the ambitions of the Linux users. Those developing server systems often opt for Red Hat or Debian. Those whose priority is easy to use desktop systems may choose SuSE or Mandrake. But because all these systems are broadly compatible, the various projects that have developed around the Linux kernel will work on each distribution. The only difference is in the installation of the software, and how you configure the system. Even so, although the exact process of configuration may vary, the principles of what you configure are pretty much the same across all distributions.

One of the key differences between distributions is the number and range of driver programs they support. Drivers allow the kernel to interact with the hardware of the computer − the processor, memory, video display, etc. For example, some distributions, such as Red Hat, support a wide range of well−developed drivers. However, SuSE supports a far wider range of drivers, but not all of them are as well developed as they might be (they may be 'beta' versions, still in development). Therefore you may find that your choice of distribution is also influenced by hardware considerations.

For example, the laptops of the Community−Linux Training Centre use Red Hat − a good server−oriented Linux distribution. But many of the materials for the CLTC project have been written on machines using SuSE because Red Hat did not have the latest drivers to support more recent, high speed desktop and laptop systems. When new versions of Red Hat come out, it is likely that drivers will be available for the newer hardware, Then the systems can be simply re−installed with Red Hat.

There is another Unix−like operating system called 'BSD' (the Berkeley Systems Distribution). This is a direct 'port' (conversion to a different machine architecture) of Berkeley Unix. It was actually developed at the same time as the original Linux kernel, but got bogged down in arguments over proprietary rights. This meant the Linux kernel took off ahead of it. BSD is not therefore a Linux−based system, but it is (with a little tinkering) able to run Linux applications because both are compliant with the 'POSIX' standard that defines Unix and similar Unix−like systems.

In the end, because of the broad compatibility of Linux−based projects, it is up to you to decide which distribution you favour. But be aware, the competition and interchange between the users of different Linux distributions verges on the tribal!

The basic principles of the Linux system

A PC computer is a blank system − it’s totally dumb. It has a basic program, called BIOS, that makes the machine power−up. But the operating system of the machine is installed by/for the user. The same machine can have Windows, or Linux put onto it − or even both if you use a system called ’dual booting’ (both are installed, but you decide which to use when you turn the machine on).

There are three important features of the Linux system that are inherited from Unix:

× ’*Everything is a file*’. This point is a little vague for the average computer user, but it has important implications. On many systems you have to worry about what the communication characteristics of the device you are working with − such as the disk drive, or the monitor, or the mouse − are. Under Linux, communication is treated as essentially writing or reading information to or from a file. This simplifies how programs work, and make it easy to implement the next feature...

× ’*The system is modular*’. As noted above, the operating system is made up of many programs that work together, each program providing a specific function. This allows greater flexibility and portability when designing programs for the system. Modularity has also assisted the development of the Linux kernel. Various Linux−based projects, because successive stages in development can be implemented as a succession of new or modified modules, enable different groups to work on different parts of the kernel. It also assists with the progressive installation and updating of programs.

× ’*Everything runs in shells*’. This last point is what makes Linux stable. Programs are assigned a shell − an allocation of resources on the computer − and the only environment that the program is aware of is the shell. Access to other parts of the system are controlled by the kernel. This means that the system has higher stability than other systems, such as *Windows* (and it’s dreaded ’blue screen of death’). ’Stability’ doesn’t mean that all the programs/applications you can use with Linux work perfectly − sometimes they don’t. What it means is that when you do get an error, the rogue program can be shut down without damaging any other programs running at that time, and without adversely impacting the Linux kernel.

The contrast between Linux and Windows then is the concept of design. Linux is a series of interlocking blocks, which provides a greater level of separation, and protection, between functions. Windows is far more monolithic − a centralised operating system where a fault in one program can propagate through the system affecting other programs, or the system’s kernel program. Linux multi−tasks programs by opening up many shells. User’s too are given access to the system from within a shell. This shell not only provides access to the functions of the system, but it can also be configured to limit or restrict access, giving different users different privileges on the system. In addition, all the files on the Linux system are caste as belong to a particular ’user’ (the *user ID number*, or *UID*), or a particular ’group’ (the *group ID number*, or *GID*). These access controls are further defined to state whether a particular user, or group, or everyone can read, write or execute a particular file. Therefore, in addition to controlling access to programs or system resources, access can also be closely controlled to files on the system. This means that if more than one user works on the same computer, they can share files together, or have ’private’ files that only they alone can access.

The added benefit of the rigid system of shells, resource allocation, and file access control not only means that the system is more stable. It also increases security. There is one user − the ’root’ user − who has complete control of the system. Other users are allocated accounts on the system with varying levels of control. This means that it us very difficult for an ordinary user to damage the operating system, delete files, or install new, unauthorised software on the system that will damage the operation of the system as a whole. It also means that computer viruses, and ’trashing programs’ (that the user is conned into running believing them to be a legitimate program), have far less impact on Linux desktop systems than on Windows desktop systems.

This segregation of users is not 100% effective. Advanced Linux users could circumvent these various controls by exploiting security flaws within the system − in order to get ’root privilege’ for example. But for most users it is sufficient to make the computer ’user proof’. For example, you can let you four−year old child loose on the computer without supervision, and not worry about what they might inadvertently do to the system that could damage it. And because user accounts are strictly segregated, your four−year old will not be able to access and corrupt or delete your own work − providing they have their own account.

Configuration, diversity, and the role of open licensing

The compatibility and interoperability of systems does not produce uniformity. Monolithic or centralised systems can. Again, this ability to have compatibility and interoperability between distributions and different Linux−based applications is directly related to the open licensing of the Linux kernel and Linux−based software.

The closed and centralised systems, such as those produced by Microsoft, are uniform. The secrets behind what makes the system work are precisely that − secrets. Only those allowed into the ’approved’ fold of Windows−based software developers have privileged access to the inner workings of the operating system. The primary bar to entry in this elite world is money − the cost of the documentation, programs and development utilities to undertake software development with Windows. This gives little option for the users to personalise or configure other than those choices granted by Microsoft, or the writers of the software that is used with the Windows system (again, predominantly Microsoft).

The need to continually upgrade, in order to keep making money, also means that closed software never has chance to settle and improve with time. It is being constant modified purely for the sake of modification, and not just to solve bugs or security flaws in the design of the program. Often, as has been seen with each new Microsoft networking or operating systems, successive releases of programs will often introduce new bugs and security flaws instead of tidying up the old ones.

By contrast, the compatibility between Linux distributions and Linux−based projects means that the same system can be configured or modified in many different ways by users. Another significant factor is that, although configuration can produce widely differing methods of programming or operation, the standards that are used within the systems are open and so everyone can use them without penalty. The fact that new releases of a system can be driven by the qualitative improvement to a system, rather than by general revision, results in genuinely better software with each new release. With open systems you are able to realise an old truism that is difficult to honour in the world of proprietary systems − "if it ain’t broke, don’t fix it".

There is also the possibility, for any Linux user who decides to undertake the effort of learning the required skills, to modify the programs that they use to suit their own particular needs. They can then, under the open license, put their particular development efforts back into the pool of code collectively owned by the Linux community.

There are also other benefits to ’open’ development that benefit wider society.

Increasingly, as part of the process of ’digital rights management’, closed software developers are incorporating ’spyware’ into their applications. These programs log the use of files, or of programs, and send this information back to the program/file developer when you next make an Internet connection. This data can then be used for a variety of rights enforcement, billing or marketing purposes, or sold to other agencies who may use it for many other purposes. Spyware can exist unchecked in proprietary/closed systems because the program code is kept secret, and often trying to unscramble/disassemble the program code is itself an infringement of the developers intellectual property rights.

Conversely, in open systems, spyware can be located, and it’s function checked to see it if endangers the privacy or security of the user. For those who object to such code being included, they could remove the code, and make the program available to other without the spyware routines included. It also does not take a huge level of programming expertise to begin interpreting the source code of a program in order to locate spyware. Unlike checking for security flaws, which requires subtle interpretation of the program’s operation, looking for suspect or unwanted code only involves assessing the operation of the different subroutines within a program.

In any case, such privacy− or security−damaging functions are unlikely to exist within open applications, unless the developer deliberately intended to include them. This is because on open systems, one of the last things that the developer is worried about controlling is the use or copying of their program!

Perhaps the greatest benefit to open development is diversity of application. Mainstream software houses only develop programs for the mass−market. The functionality contained in those programs will also be skewed towards the major needs of mass−market users. This inevitably means the software produced will meet the majority of users needs, but not all. However, because the utilities required to develop new applications are included as standard with Linux distributions, anyone has the ability to develop software that meets their own needs. More importantly, using online services − such as the *Sourceforge* or *Freshmeat* web sites −users with a common ’minority’ interest can work collaboratively to develop new applications. This is because collectively they have the equivalent time and resources of the mainstream software development corporations.

Choice and the ’information society’

This briefing has sought to outline a little about what Linux is, and how it works. There is one final aspect of using Linux that we should all be aware of − *choice*. From the work of Richard Stallman on, there is an important emphasis within the development of open systems that looks to the impacts technological systems have on society. One of the key issues is that if this new information society is to work for all, then we its citizens must be included within the forces that shape its development.

The new information society, being forged by greater technological complexity and global data networks, has implications for our economic future. But it also has serious implications for human rights, and new forms of inequality and exploitation enabled by technological systems. Under the old industrial revolution, the technology of new inventions was ’immediate’. Development was local. People could be involved with local development, and either work for or against it. This is not true within the information revolution. Networks provide distance, and without equal access to networks, that distance can provide a buffer between power and the democratic process.

We face a choice in how the information society develops.

We can continue to maintain the legally−sanctioned privateering that intellectual property rights confer. In a world where all activities will become increasingly mediated by technology, the effect of this will be to further isolate the people from the control of technological progress. We've seen this already in other areas, such as the debate over genetically modified food − industry and society of opposing courses.

The alternative is for society to consider precisely what levels of intellectual property are consistent with a free and fair society. Linux, and the development of 'open' systems networks, is an important part of this process. It enables the standards that underpin the operation of systems to be owned in common by the people who use those systems. In this way, the types of inequality caused by closing off networks with the financial or legal blocks enabled by intellectual property cannot have the same power within open systems as within proprietary ones.

Already, in the United States, the corporations that have an interest in increasing intellectual property rights are funding politicians to introduce new laws. Some of these, like the Digital Millennium Copyright Act (DMCA), have been highly damaging to the computing community − in particular the 'open source' developers. Most of the proposals, such as 'chipping' every device to track the use of digital information, or allowing copyright holders to hack peer−to−peer networks sharing information, go much further than the DMCA. It should be noted that such legally sanctioned '*cyber−vigilante*' arguments are not some recent pipe−dream. They have been the subject of books and papers for at least the last ten years within the US (e.g., *Netspionage*).

What we are seeing in these new proposals is the legislature of the US giving economic rights to certain groups in society that they can exert over society as a whole. To make this power realistic, these new proposals include extra−judicial powers for property owners to take action. Of course, the US has the most highly developed intellectual property laws in the world. But what happens there is only a year two away from happening here.

In conclusion, do the world, and the development of the new information society a favour. Dump closed systems. It is a preposterous notion that what drives the mass software market is consumer demand. Microsoft have been selling flawed software for years, and the public have been unable to anything about it. Now that Linux has begun to provide better support for desktop computing, that is no longer the case.

Unless we seek to control the power of proprietary systems soon, it may be too late. If we can develop a critical mass of the public supporting open systems, then it will not be politically possible for the proponents of a wholly closed technical infrastructure to succeed.

Links

'*The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*', Eric Raymond (O'Reilly Associates 1999) − online at http://www.tuxedo.org/~esr/writings/cathedral−bazaar/

Slashdot − http://www.slashdot.org/

The GNU Project − http://www.gnu.org/

Free Software Foundation − http://www.fsf.org/

Linux Links − The Linux Portal Site − http://www.linuxlinks.com/

Linux Online − http://www.linux.org/

Linux Newbie (info. for beginners) − http://www.linuxnewbie.org/

'*The GreenNet Internet Rights Toolkit*' − http://www.internetrights.org.uk/ or http://www.fraw.org.uk/v_library/section_a/rangers_pws.shtml

Hard copy

'*Rebel Code − Linux and the Open Source Revolution*', Glyn Moody (Allen Lane/Penguin Press 2001)

'*The Hacker Ethic and the Sprit of the Information Age*', Pekka Himanen (Seeker and Warburg 2001)

'*Business at the Speed of Thought*', Bill Gates (Penguin Books 1999)

'*Netspionage and the Global Threat to Informatio*n', Boni and Kovacich (Butterworth−Heinemann 2000)

'*The Ages of Access − how the shift from ownership to access is transforming modern life*', Jeremy Rifkin (Penguin Books 2000)