The Gnu/Linux System

The alternative to proprietary computer systems

Sheet **J.2**

Version 1.1, March 2009. Produced by the Free Range *Community–Linux Training Centre* (CLTC) Project
http://www.fraw.org.uk/cltc/ cltc@fraw.org.uk

30p (where sold)

From the point of view of Windows users, where you don't necessarily have to understand what you're doing to make the machine work, Gnu/Linux can seem a complicated and difficult thing to get your head around. In reality, whilst you might have to become a little more "involved" in using your computer, the benefits – especially better security, lower costs and far less vulnerability to viruses – will outweigh the costs of learning to use a different system.

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

In the beginning, there was *Unix*...

...and Unix[54] was developed as a *modular, networked, multi-tasking, multi-user operating system.* It was also "platform independent" as it was coded in a standardised programming language – *C* – that allowed programs to be shared on different types of computer system.

"Multi-Tasking"[1] meant that it could run many processes (programs) at the same time; *"Multi-User"*[2] meant that more than one person could use it at the same time; and an *"Operating System"*[3] is a collection of programs that make the computer work, carrying out essential tasks and running communications between the electronics of the computer and the programs that you are running on it.

"Modular"[4] 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. The benefits of this is that, unlike a complex monolithic operating system (like Windows), it's easier to find bugs and upgrade the system.

"Networked"[5] essentially means that Unix is designed to operate across an environment of connected machines, or acting as a server for a number of client machines. Unlike other systems, where networking is an add-on, Unix was designed from the outset to run functions that could operate over a number of computer systems. In the beginning the network would have been "dumb terminals" (just a screen and a keyboard – they don't do any processing) where the employees of a large corporation or the students at a university would use the functions of the computer. But it could also be another Unix server located in another building, or another country, allowing communications to be passed between machines (the Internet arose at the same time as Unix-based networks were developed).

"C"[6] is a programming language. Then, with the use of a simple compiler program[7] that's specific to the system or processor that the program is to run on, you create programs from the C code. This was revolutionary because until that time programs were mostly written for specific computers rather than having a standard language that could be interpreted for use on different types of computer.

The fact that compilers were easily available also meant that anyone able to read the code could make their own modifications and build them into the operating system on their machine – and then pass-on those changes to others (*which was of course the most important aspect of the later development of Gnu/Linux programs*).

Today, now 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" (see below) who were just beginning to play with microcomputers, there were two problems with Unix. Firstly, Unix would not run on a system with the minuscule resources of early microcomputers. Secondly, Unix belonged to someone – *it wasn't free for anyone to use and play with.*

"Hackers" are not (necessarily) bad people!

At various points in this handout we use the term "hacker". The media always portray hackers as people who break into computer systems and write viruses. In reality the term "hacker"[8] was developed in the late 60s to describe a person who was good at playing around with technology and making it do new things.

When the media use the phrase "hacker" what they should say is "cracker"[9] – a person who cracks the security and exposes the flaws of computer/security systems. Whilst most crackers will to some extent be hackers, it doesn't follow that all hackers are crackers (although some hackers are completely crackers and should try to get outside more!)

Unix for the PC!

There were a number of attempts to re-create Unix for the IBM-compatible PC. In the early 1980s Santa Cruz Operations and Microsoft collaborated to produce a PC-version of Unix, called *Xenix*[10]. 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 to produce a free Unix-like operating system, called *Minix*[11], but again it never took off because of the problems of developing Minix as a fully-functioning operating system.

The late 1980s saw an explosion in the use of the Internet at universities. 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*[12] (Microsoft Disk Operating System). However, the Internet allowed a new way of developing computer programs; many individual computer enthusiasts with a common interest could work collaboratively to develop a new project. One computer hacker in particular was able to grasp this concept – Richard Stallman.

GNU/Linux is born

Richard Stallman[13] is the founder of the Free Software Foundation[14] and the initiator of the GNU Project[15] (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*[16] – 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*[17] – the core of the operating system that runs the machine. At this time a 21-year old Finnish computer student, Linus Torvalds[18], bought an IBM PC. He used Microsoft's MS-DOS for a while, but shortly after obtained a copy of Minix and started playing with it. This was possible because Stallman's GNU-licensed software ran on Minix (all Unix's work in a similar way).

After playing with Minix for a few months Torvalds embarked on writing his own small operating system. Rather than doing it all himself he invited others to join in. This meant that rather than relying on his own resources and abilities the project could be developed by involving many people, with many different talents, who could between them wield the time and resources of a large computer corporation (the principle that still underpins many of the Linux-based projects under-way today). Between mid-1991 and early 1994 the Linux OS developed from a minute kernel to a working Unix-like operating system. The Linux kernel[19] 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 live, and began replicating its way across the Internet – gathering more users, spawning new projects to improve the kernel as well as new applications that used the kernel.

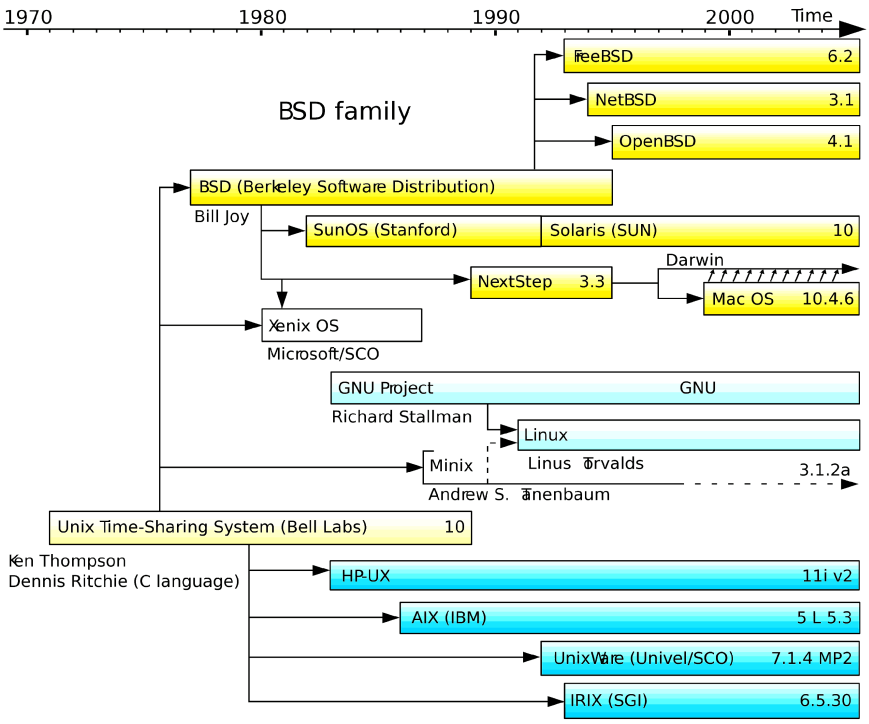
Today, Linus Torvalds still guides the development of the kernel in collaboration with many kernel hackers across the globe. At the same time other projects have grown up around the development of the kernel that are equally significant – and all symbiotically related like the original complementary work of Stallman and Torvalds. For example, the *Xfree86 Project*[20] 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*[21] (GUIs) such as GNOME[22] or KDE[23]. 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 functions amongst the Gnu/Linux community.

Linux grows and develops "distributions"

Linux is not a complete operating system; it's just the kernel that, like a conductor of an orchestra, organises what happens inside the computer. An operating system requires a kernel plus a large number of other programs and utilities, which perform specific tasks, to make it a viable *system* for the computer users. For this reason different groups have developed Linux *distributions*[24], each providing a slightly different set of applications to the others – although most contain a standard set of widely used applications meaning that they are compatible with each other.

There are a growing number of distributions available, but there are a few that are widely used by most Gnu/Linux users – e.g., Red Hat[25], Fedora[26], SuSE[27], Debian[28], Slackware[29] and Ubuntu[30]. Then there are many others that have evolved to suit specialised needs – such as secure information servers. Some systems that appear to be one thing, such as a stand alone firewall system like Smoothwall[31], are actually a Linux distribution that works solely as a firewall and nothing else. Others, such as Knoppix[32], run "live" from a CD or DVD so that you don't have to install the Gnu/Linux system on the machine to use a Gnu/Linux system.

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. One of the main differences between distributions is the number and range of device driver programs[34] 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/Fedora, support a wide range of well-developed drivers, but SuSE supports a far wider range of drivers although not all of them are as good as they might be (they may be "beta versions", still in development). Conversely Fedora only includes "free software", and for this reason it lacks a number of the drivers used in other distributions that contain proprietary code. Therefore you may find that your choice of distribution is also influenced by hardware considerations.



The Unix 'family tree'[33]

There is another PC-based Unix-like operating system called BSD[35] (the Berkeley Systems Distribution), which is a version of Berkeley Unix developed for the PC (coincidentally, it was also the root of what became *OS X*[36], the Apple Macintosh operating system). It was developed at the same time as the Linux kernel but got bogged down in arguments over proprietary rights, and so Linux 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. This is because both are compliant with the POSIX standard[37] that defines the structure of 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 beware!* – 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, in logic terms, an empty box – *it's totally dumb*. It has a basic program held on a computer chip called the *Basic Input-Output System*[38] (BIOS). This makes the machine power-up, and load whatever operating system is on the storage system. The same machine can have Windows or Gnu/Linux put onto it – or even both if you use a system called dual booting[39] (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 the structure or the communication characteristics of the device that you are working with – such as the disk drive, or the monitor, or the mouse. Under Linux communication is treated as essentially writing or reading information to or from a file. This simplifies how programs work, and make it easier 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. Various Linux-based projects, because successive stages in development can be issued as new or modified modules, progressively grow the functions of the program over time. Modularity also assists with the checking, installation and updating of programs by the system.

Everything runs in 'shells'. This last point is what makes Linux stable. Programs are assigned a shell[40] – an allocation of resources on the system. The program is only aware of its shell and so its access to other parts of the system can be controlled by the kernel. This means that the system has better stability, although this 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 affecting the Linux kernel.

The main difference between Gnu/ Linux and

Windows is *design*. Linux is a series of interlocking blocks, which provides a greater level of separation, and protection, between functions. Windows is a 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. These shells not only provide access to the functions of the system but they can also be configured to limit or restrict access, giving different users different levels of access to the system. In addition, all the files on the Linux system are caste as belong to a user (the *user ID number*, or UID), or group (the *group ID number*, or GID). File access controls also state whether a user, group, or anyone, can read, write or execute a particular file. Therefore, in addition to controlling access to programs or system resources, access to files on the system can also be controlled. 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 can access.

This 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 only one user – *the root user* – who has complete control of the system. Other users, including certain programs that provide services (like web servers), are allocated accounts on the system with varying levels of control. This means that it is very difficult for an ordinary user, or a program, to damage the operating system, delete files, or install new, unauthorised software on the system. This prevents damage to the operation of the system as a whole, but it also means that computer viruses and other malware[41] have far less of an impact on Linux systems than on Windows systems.

To date there have been no serious Linux malware[42] incidents, of the type that Windows regularly suffers from, because of the malware program's lack of root access. Also, the more regular updating of each Linux distribution means that any security holes that are found can be fixed very quickly.

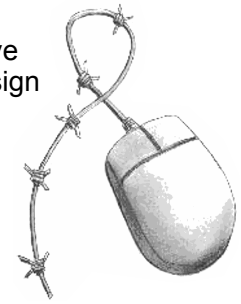
This segregation of users is not 100% effective. Advanced Linux users could circumvent these various controls by exploiting security flaws within the system in order to get root privilege, but for most users it is sufficient to make the computer “user proof”. For example, you can let your six-year old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

Configuration, diversity, and open licensing

The compatibility of Gnu/Linux systems does not produce uniformity – just the opposite. It encourages a diversity of uses because information can be reliably exchanged using common standards. Proprietary systems enforce uniformity, through copyright or software patents that restrict the right of others to modify the functions of the system, in order to restrict compatibility between competing systems. The ability to have a high level of compatibility and interoperability between different Gnu/Linux distributions, and the wide range of Linux-based applications, is directly related to the open licensing of the Linux kernel and Gnu/Linux-based software.

The closed and centralised systems, such as those produced by Microsoft and other proprietary software developers, are uniform because modifications to the code of the system are not just discouraged, they're legally barred. The secrets behind what makes the system work are precisely that – *secrets*. For example, only those allowed into the approved fold of Windows-based software developers have privileged access to the inner workings of the operating system (such as Microsoft's *shared source*[43] method of code distribution). The primary bar to entry in this elite world is of course 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 the system, other than those choices granted by Microsoft, or the writers of the software that is used with the Windows system (again predominantly, but not exclusively, Microsoft).

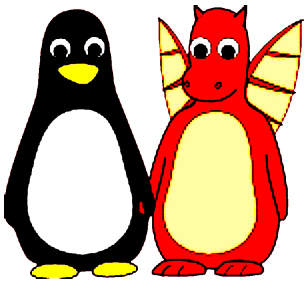
The need to continually upgrade to a wholly new version of the system, in order to keep the software producers' income flowing, also means that closed software never has chance to settle and improve with time. It is being constantly modified purely for the sake of modification, and not just to solve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



By contrast, the compatibility between Gnu/Linux projects allows the code produced by one developer to be used by others. This means that once good, secure code has been written it can be reused widely rather than producing new, insecure code. Ultimately this means that new code releases can be driven by the qualitative improvement to a system, rather than by a business-oriented general revision, and produces genuinely better software with each new release (with open licensing you are able to realise one of the most important concepts in engineering – "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. All the documentation, program code, and the programming tools you need to do the work, are provided as standard with most Gnu/Linux distributions. Those with an interest in playing around with the code can then, under the freedom created by the open license, put their particular development efforts back into the pool of code collectively owned by the Gnu/Linux community.

One of the strengths of this approach hasn't just been the freedom to modify the code, but also the human languages that the programs can handle. Through a process known as *localisation*[44] Gnu/Linux programs have been translated into many languages, including a number of minority languages which most large software companies do not support (yes, Gnu/Linux even comes in Welsh[45] and Gaelic[46] versions!)



There are also other benefits to 'open' development that benefit wider society. Increasingly, as part of the process of digital rights management[47], closed software developers are incorporating spyware[48] into their applications. These programs log the use of files, or 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 easily exist, undiscovered, in proprietary/closed systems because the program code is kept secret. Even trying to unscramble/disassemble the program code is of itself an infringement of the developers intellectual property rights. This makes the detection and blocking or removal of spyware from proprietary programs very difficult.

The greatest benefit of open development is “diversity”. Mainstream software houses develop programs for the mass-market and so the programs are skewed towards the needs of the major users – *large businesses*. This inevitably means the software produced might meet the majority of a users needs, but only if you pay for the “premium edition” used by large businesses. Of course, you could buy software development programs, like those that come with Gnu/Linux systems, but they are very expensive – and so software development is restricted to those with the ability to pay rather than those with the ability to code.

With Gnu/Linux, because the utilities required to develop new applications are included as standard with most distributions, anyone has the ability to develop software that meets their own needs. In fact, most Gnu/Linux distributions come with a variety of programming languages such as *C*, *Perl*, *Python*, and *Java*. Then, using on-line services – such as *Sourceforge*[49] or *Freshmeat*[50] – those with a common interest can work collaboratively to develop programs. 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 Gnu/Linux is, and how it works. But there is one aspect of its use that we must all cherish – *choice*. From the work of Richard Stallman onward, there is has been an emphasis within the development of open systems on considering the impacts that technological systems have on society. If the new “Information Society” is to work for all, then all its citizens must be included within the processes that shape its development. Otherwise we just reinforce the divisions of the old Industrial Society using the power of information technology.

Another aspect related to our ability to choose is our ability to pay. The table on the next page provides a “cost comparison” of installing different types of operating system on a computer. Clearly, Gnu/Linux wins out when we consider the cost implications of our system selection. This illustrates one of the motivations behind this project, and the choice that we all face over how the information society develop in the future:

- We can continue to maintain the legally-sanctioned privateering that intellectual property rights confer in the digital domain but, 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.
- The alternative is for society to consider precisely what levels of intellectual property are consistent with a free and fair society.

Gnu/Linux, and the development of open systems, is an important part of the process of making a technologically mediated society accessible to all. 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 technology with financial or legal blocks, enabled by intellectual property, cannot take place.

The idea that there is an area of society that belongs to no one, *a right of common*, has existed in many states throughout history. Open technology enables up to take this same concept into the Information Age. Unless we seek to control the power of proprietary systems soon it may be too late to do so. If we can develop a critical mass of the public supporting open systems, with the skills required to support this approach, then it will not be politically possible for the proponents of a wholly closed technical infrastructure to succeed.

A cost comparison of Gnu/Linux and Microsoft system software

Software	Gnu/Linux (via Linux Emporium[51])		Windows XP (via PC World[52])		Windows Vista (via PC World[52])	
Operating System	Ubuntu 8.1	£5.91	XP Home*	£79.99	Home Premium	£166.95
Office Suite	OpenOffice	*Free*	Office 2007 Home	£99.95	Office 2007 Home	£99.95
Graphics Editor	GIMP	*Free*	Paint Shop Pro	£58.95	Paint Shop Pro	£58.95
Desktop Publisher	Scribus	*Free*	Microsoft Publisher	£99.95	Microsoft Publisher	£99.95
Web Design	*various utilities*	*Free*	Magix Website	£29.95	Magix Website	£29.95
CD/DVD Burning	K3B	*Free*	Nero9	£39.96	Nero9	£39.96
Anti-Virus	*various utilities*	*Free*	Norton360	£58.71	Norton360	£58.71
Data Backup	*various utilities*	*Free*	Acronis Home 2009	£39.95	Acronis Home 2009	£39.95
TOTAL		**£5.91**		**£507.41**		**£594.37**

All prices include VAT – prices obtained on 21/3/09 **not from PC World (who don't sell XP), price via AUT Direct[53]*

This is not intended to be a "prefect" comparison, as this is difficult in any case because of the slightly different functionality of the different programs involved. What this comparison assumes that you have a blank computer and you wish to install a variety of software to produce a general purpose home/office computer system. The comparison also doesn't take account of the difference in performance between these operating systems on older computer hardware, and the impact this has upon the cost of buying the machine: Vista only works on the latest computer hardware; XP will only work well on fairly powerful systems; but theoretically you can install Gnu/Linux on a cheap five- or six-year old computer, provided you put in enough memory to make the speed usable.

In conclusion, do yourself and the world a favour – *dump your closed software*!

It is a preposterous notion that what drives the software market is "consumer demand" – what in fact drives the market is corporate computing, but mostly the opportunities for financial expropriation created the restrictions of intellectual property rights. Microsoft have been selling flawed software for years, and the public have been unable to do anything about it. Now that Gnu/Linux provides far better support for desktop computing this is no longer the case.

Web References

1. http://en.wikipedia.org/wiki/Computer_multitasking
2. http://en.wikipedia.org/wiki/Multiuser
3. http://en.wikipedia.org/wiki/Operating_system
4. http://en.wikipedia.org/wiki/Modularity_(programming)
5. http://en.wikipedia.org/wiki/Computer_network
6. http://en.wikipedia.org/wiki/C_(programming_language)
7. http://en.wikipedia.org/wiki/Compiler
8. http://en.wikipedia.org/wiki/Hacker_(programmer_subculture)
9. http://en.wikipedia.org/wiki/Hacker_(computer_security)
10. http://en.wikipedia.org/wiki/Xenix
11. http://en.wikipedia.org/wiki/Minix
12. http://en.wikipedia.org/wiki/MSDOS
13. http://en.wikipedia.org/wiki/Richard_Stallman
14. http://www.fsf.org/
15. http://www.gnu.org/
16. http://en.wikipedia.org/wiki/Gpl
17. http://en.wikipedia.org/wiki/Kernel_(computing)
18. http://en.wikipedia.org/wiki/Linus_Torvalds
19. http://en.wikipedia.org/wiki/Linux_kernel
20. http://en.wikipedia.org/wiki/Xfree86
21. http://en.wikipedia.org/wiki/Graphical_user_interface
22. http://en.wikipedia.org/wiki/GNOME
23. http://en.wikipedia.org/wiki/KDE
24. http://en.wikipedia.org/wiki/Linux_distribution
25. http://www.redhat.com/
26. http://www.fedoraproject.org/
27. http://www.opensuse.org/en/
28. http://www.debian.org/
29. http://www.slackware.com/
30. http://www.ubuntu.com/
31. http://www.smoothwall.org/
32. http://www.knopper.net/knoppix/index-en.html
33. http://en.wikipedia.org/wiki/File:Unix_history.en.svg
34. http://en.wikipedia.org/wiki/Device_driver
35. http://www.freebsd.org/
36. http://en.wikipedia.org/wiki/OSX
37. http://en.wikipedia.org/wiki/POSIX
38. http://en.wikipedia.org/wiki/BIOS
39. http://en.wikipedia.org/wiki/Dual_boot
40. http://en.wikipedia.org/wiki/Shell_(computing)
41. http://en.wikipedia.org/wiki/Malware
42. http://en.wikipedia.org/wiki/Linux_malware
43. http://en.wikipedia.org/wiki/Shared_source
44. http://en.wikipedia.org/wiki/Internationalization_and_localization
45. http://www.cymrux.org.uk/
46. http://borel.slu.edu/gnu/
47. http://en.wikipedia.org/wiki/Digital_rights_management
48. http://en.wikipedia.org/wiki/Spyware
49. http://sourceforge.net/
50. http://freshmeat.net/
51. http://www.linuxemporium.co.uk/
52. http://www.pcworld.co.uk/
53. http://www.autdirect.co.uk/
54. http://en.wikipedia.org/wiki/Unix

Produced by the Free Range 'Community–Linux Training Centre' (CLTC) Project – http://www.fraw.org.uk/cltc/