Chapter 1

1. UNIX

Unix (often spelled "UNIX," especially as an official trademark) is an operating systemthat originated at Bell Labs in 1969 as an interactive time-sharing system. Ken Thompson and Dennis Ritchie are considered the inventors of Unix. The name (pronounced YEW-nihks) was a pun based on an earlier system, Multics. In 1974, Unix became the first operating system written in the C language. Unix has evolved as a kind of large freeware product, with many extensions and new ideas provided in a variety of versions of Unix by different companies, universities, and individuals.

Partly because it was not a proprietary operating system owned by any one of the leading computer companies and partly because it is written in a standard language and embraced many popular ideas, Unix became the first open or standard operating system that could be improved or enhanced by anyone. A composite of the C language and shell (user command) interfaces from different versions of Unix were standardized under the auspices of the IEEE as the Portable Operating System Interface (POSIX). In turn, the POSIX interfaces were specified in the X/Open Programming Guide 4.2. These interfaces are also known as the "Single UNIX Specification" and, in the most recent version, "UNIX 03"). The trademarked "UNIX" is now owned by the The Open Group, an industry standards organization, which certifies and brands Unix implementations.

Unix operating systems are used in widely-sold workstation products from Sun Microsystems, Silicon Graphics, IBM, and a number of other companies. The Unix environment and the client/server program model were important elements in the development of the Internet and the reshaping of computing as centered in networks rather than in individual computers. Linux, a Unix derivative available in both "free software" and commercial versions, is increasing in popularity as an alternative to proprietary operating systems.

Linux - Unix Differences in Cost & Distribution

Linux can be freely distributed, as it is an open Source OS. So anyone can get a copy of Linux from books, magazines, or from the internet also. For server versions,organizations typically pay distributors for a support contract, not the software. The major distributors are RED HAT, Mandrake, and SUSE. For server hardware, IBM, HP,Dell are the major ones.

UNIX is costly as compared to Linux; the midrange UNIX servers are priced in between $25,000 and $249,999 (including hardware). The major distributors are HP, IBM and SUN. A high end UNIX server can cost up to $500,000. According to IDC, Gartner, IBM is the market leader in UNIX servers, HP is in 2nd position and SUN is in the third position.

Commercial UNIX is usually custom written for each system, making the original costquite high, whereas Linux has base packages also. In this respect, Linux is closer in its model to Windows than a commercial UNIX OS is. At the time of purchasing a UNIX server, users get a Vendor assistance plan on setting up and configuring the system. But with Linux, Vendor support must be purchased separately.

Threats and Security: Unix vs. Linux

Both of the operating systems are vulnerable to bugs but Linux is far more responsive in dealing with the threats. Linux incorporated many of the same characteristics and functions found in UNIX, including the segmentation of the user domain in a multi-user environment, the isolation of tasks in a multi-tasking environment, a password system that can be encrypted and/or located remotely and much more. As Linux is an open system OS, the bugs can be reported by anyone in the user/developers forum, and within days it can be fixed. But for UNIX, this is not the case, and user has to wait for a while, to get the proper bug fixing patch. The open source community delivers faster because it does not have to go through the endless development cycles of commercial-based operating systems.

At the same time, as an open source operating system, it is supported by tens of thousands of developers worldwide. To reiterate, this allows for better innovation and quicker-to-market features than anything UNIX can provide.

Market and future of Linux and Unix

According to International Data Corp. (IDC).Linux has grown faster than any other server OS over the past few years. Linux user base is estimated to be about more than 25 million machines, compared to 5.5 million for combined UNIX installations.
Linux is gaining popularity because of its application in embedded technologies, free and easily availability. To compete with Linux, vendors such as HP, IBM, Sun are making customized UNIX with graphical user interface and user friendly interface which is also compatible with Linux. The main UNIX vendors--IBM, Sun, and Hewlett-Packard are already putting Linux interoperability features into future releases of AIX, Solaris, and HP-UX.

2. Kernel

A Unix kernel — the core or key components of the operating system — consists of many kernel subsystems like process management, scheduling, file management, device management and network management, memory management, dealing with interrupts from hardware devices.

Each of the subsystems has some features:

Concurrency: As Unix is a multiprocessing OS, many processes run concurrently to improve the performance of the system.[disputed – discuss]

Virtual memory (VM): Memory management subsystem implements the virtual memory concept and users need not worry about the executable program size and the RAM size.[disputed – discuss]

Paging: It is a technique to minimize the internal as well as the external fragmentation in the physical memory.

Virtual file system (VFS): A VFS is a file system used to help the user to hide the different file systems complexities. A user can use the same standard file system related calls to access different file systems.

The kernel provides these and other basic services: interrupt and trap handling, separation between user and system space, system calls, scheduling, timer and clock handling,file descriptor management.

3. The Shell

A Unix shell is a command-line interpreter or shell that provides a traditional user interface for the Unix operating system and for Unix-like systems. Users direct the operation of the computer by entering commands as text for a command line interpreter to execute, or by creating text scripts of one or more such commands. Users typically interact with a Unix shell using a terminal emulator, however, direct operation via serial hardware connections, or networking session, are common for server systems.

What is "the shell"?

Simply put, the shell is a program that takes your commands from the keyboard and gives them to the operating system to perform. In the old days, it was the only user interface available on a Unix computer. Nowadays, we have graphical user interfaces (GUIs) in addition to command line interfaces (CLIs) such as the shell.

On most Linux systems a program called bash (which stands for Bourne Again SHell, an enhanced version of the original Bourne shell program, sh, written by Steve Bourne) acts as the shell program. There are several additional shell programs available on a typical Linux system. These include: ksh, tcsh and zsh.

What's an xterm, gnome-terminal, konsole, etc.?

These are called "terminal emulators." They are programs that put a window up and let you interact with the shell. There are a bunch of different terminal emulators you can use. Most Linux distributions supply several, such as: xterm, rxvt, konsole, kvt, gnome-terminal,nxterm, and eterm.

Starting a Terminal

Your window manager probably has a way to launch programs from a menu. Look through the list of programs to see if anything looks like a terminal emulator program. In KDE, you can find "konsole" and "terminal" on the Utilities menu. In Gnome, you can find "color xterm," "regular xterm," and "gnome-terminal" on the Utilities menu. You can start up as many of these as you want and play with them. While there are a number of different terminal emulators, they all do the same thing. They give you access to a shell session.

4. The init process

In Unix-based computer operating systems, init (short for *initialization*) is the first process started during booting of the computer system. Init is a daemon process that continues running until the system is shut down. It is the direct or indirect ancestor of all other processes and automatically adopts all orphaned processes. Init is started by the kernel using a hard-coded filename; a kernel panic will occur if the kernel is unable to start it. Init is typically assigned process identifier 1.

The kernel, once it is loaded, finds init in sbin and executes it.

When init starts, it becomes the parent or grandparent of all of the processes that start up automatically on your Linux system. The first thing init does, is reading its initialization file, /etc/inittab. This instructs init to read an initial configuration script for the environment, which sets the path, starts swapping, checks the file systems, and so on. Basically, this step takes care of everything that your system needs to have done at system initialization: setting the clock, initializing serial ports and so forth.

Then init continues to read the /etc/inittab file, which describes how the system should be set up in each run level and sets the default *run level*. A run level is a configuration of processes. All

UNIX-like systems can be run in different process configurations, such as the single user mode, which is referred to as run level 1 or run level S (or s). In this mode, only the system administrator can connect to the system. It is used to perform maintenance tasks without risks of damaging the system or user data. Naturally, in this configuration we don't need to offer user services, so they will all be disabled. Another run level is the reboot run level, or run level 6, which shuts down all running services according to the appropriate procedures and then restarts the system.

Use the who to check what your current run level is:

```
willy@ubuntu:~$ who -r
         run-level 2 2006-10-17 23:22                   last=S
```

After having determined the default run level for your system, init starts all of the background processes necessary for the system to run by looking in the appropriate rc directory for that run level. init runs each of the kill scripts (their file names start with a K) with a stop parameter. It then runs all of the start scripts (their file names start with an S) in the appropriate run level directory so that all services and applications are started correctly. In fact, you can execute these same scripts manually after the system is finished booting with a command like /etc/init.d/httpd *stop* or service *httpd stop* logged in as *root*, in this case stopping the web server.

fter init has progressed through the run levels to get to the default run level, the /etc/inittab script forks a getty process for each virtual console (login prompt in text mode). getty opens tty lines, sets their modes, prints the login prompt, gets the user's name, and then initiates a login process for that user. This allows users to authenticate themselves to the system and use it. By default, most systems offer 6 virtual consoles, but as you can see from the inittab file, this is configurable.

The **inittab** file describes which processes are started at bootup and during normal operation (e.g. /etc/init.d/boot, /etc/init.d/rc, gettys...). **Init**(8) distinguishes multiple *runlevels*, each of which can have its own set of processes that are started. Valid runlevels are **0**-**6** plus **A**, **B**, and **C** for **ondemand** entries. An entry in the **inittab** file has the following format:

id:*runlevels*:*action*:*process*

Special configuration in `/etc/inittab`

The `/etc/inittab` has some special features that allow **init** to react to special circumstances. These special features are marked by special keywords in the third field. Some examples:

`Powerwait`

> Allows **init** to shut the system down, when the power fails. This assumes the use of a UPS, and software that watches the UPS and informs **init** that the power is off.

`ctrlaltdel`

> Allows **init** to reboot the system, when the user presses ctrl-alt-del on the console keyboard. Note that the system administrator can configure the reaction to ctrl-alt-del to be something else instead, e.g., to be ignored, if the system is in a public location. (Or to start **nethack**.)

`sysinit`

> Command to be run when the system is booted. This command usually cleans up `/tmp`, for example.

The list above is not exhaustive. See your `inittab` manual page for all possibilities, and for details on how to use the above ones.

Linux Logging in, Logging Out, and Shutting down

Logging in

Once you have completed your system install and booted your system, you should see a login prompt on your monitor. When you did your Linux install you should have set a root password. You may have also created a user with a password. Therefore to log in, you will want to type the name of a user or "root" for the login name and enter the appropriate password. If you logged in as a normal user and know the root password and want to use administration commands, you may use the command "su" to become a "super user". Some systems also support the "sudo" command, which allows administrative privileges on a command by command basis.

Linux Shell levels and the su command

The command, "su" will allow a normal user to enter a new shell level as the root user or as another user if they know the root user's or that user's password respectively. To become the root user, type "su" then you will be prompted for the root password. To become another user, type "su username". You must enter either that user's password to become that user. Every time you use the su command you enter a new shell level which means you have invoked a new running copy of the shell program, such as bash. You can see this change by typing the command "env" and looking at the value of the environment variable "SHLVL". This value increments when you use the su command and decrements when you use the "exit" command to exit that shell environment. You can also see the shell level value by typing "printenv SHLVL".

Logging out

Use the command "logout" to exit a given session. If you have logged in, then typed "su" to become a superuser or another user, you may need to type "exit" until your SHLVL environment value is 1. Then you can type "logout" to exit your session. The "exit" command will take you back to previous shell levels.

Shutting Linux Down

The system is intended to be shutdown by the system administrator using the shutdown command in one of the forms shown below. Many systems are set up to capture the <CTRL><ALT><DEL> keystroke combination to issue the shutdown command through the init program. This will work on most systems if the root user is logged in. Examples of using the shutdown command are shown below.

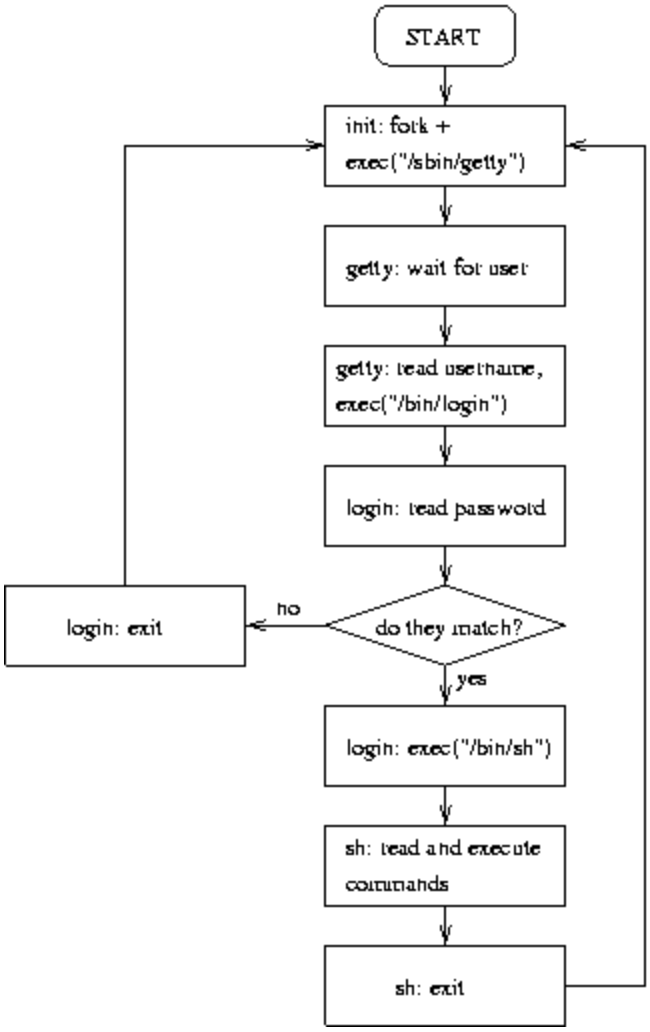
```
shutdown -h now
shutdown -r +10 "Rebooting in 10 minutes"
shutdown -r 13:00
```

The first command will shutdown and halt the system immediately. The second will reboot the system in 10 minutes and send the message to all users. The third command will shut the system down and do a reboot at 1:00 in the afternoon.

Logins via terminals

Fig. shows how logins happen via terminals. First, **init** makes sure there is a **getty** program for the terminal connection (or console). **getty** listens at the terminal and waits for the user to notify that he is ready to login in (this usually means that the user must type something). When it notices a user, **getty** outputs a welcome message (stored in `/etc/issue`), and prompts for the username, and finally runs the**login** program. **login** gets the username as a parameter, and prompts the user for the password. If these match, **login** starts the shell configured for the user; else it just exits and terminates the process (perhaps after giving the user another chance at entering the username and password). **init** notices that the process terminated, and starts a new **getty** for the terminal.

Figure 10-1. Logins via terminals: the interaction of init, getty, login, and the shell.



Note that the only new process is the one created by **init** (using the `fork` system call); **getty** and **login** only replace the program running in the process (using the `exec` system call).

A separate program, for noticing the user, is needed for serial lines, since it can be (and traditionally was) complicated to notice when a terminal becomes active. **getty** also adapts to the speed and other settings of the connection, which is important especially for dial-in connections, where these parameters may change from call to call.

There are several versions of **getty** and **init** in use, all with their good and bad points. It is a good idea to learn about the versions on your system, and also about the other versions (you could use the Linux Software Map to search them). If you don't have dial-ins, you probably don't have to worry about **getty**, but **init** is still important.

Logins via the network

Two computers in the same network are usually linked via a single physical cable. When they communicate over the network, the programs in each computer that take part in the communication are linked via a*virtual connection*, a sort of imaginary cable. As far as the programs at either end of the virtual connection are concerned, they have a monopoly on their own cable. However, since the cable is not real, only imaginary, the operating systems of both computers can have several virtual connections share the same physical cable. This way, using just a single cable, several programs can communicate without having to know of or care about the other communications. It is even possible to have several computers use the same cable; the virtual connections exist between two computers, and the other computers ignore those connections that they don't take part in.

That's a complicated and over-abstracted description of the reality. It might, however, be good enough to understand the important reason why network logins are somewhat different from normal logins. The virtual connections are established when there are two programs on different computers that wish to communicate. Since it is in principle possible to login from any computer in a network to any other computer, there is a huge number of potential virtual communications. Because of this, it is not practical to start a **getty** for each potential login.

There is a single process inetd (corresponding to **getty**) that handles all network logins. When it notices an incoming network login (i.e., it notices that it gets a new virtual connection to some other computer), it starts a new process to handle that single login. The original process remains and continues to listen for new logins.

To make things a bit more complicated, there is more than one communication protocol for network logins. The two most important ones are **telnet** and **rlogin**. In addition to logins, there are many other virtual connections that may be made (for FTP, Gopher, HTTP, and other network services). It would be ineffective to have a separate process listening for a particular type of connection, so instead there is only one listener that can recognize the type of the connection and can start the correct type of program to provide the service. This single listener is called **inetd**

The tools and application:

A tool is a simple program, usually designed for a specific purpose, it is sometimes referred as a command.

The " Unix tools philosophy" emerged during the creation of the UNIX operating system, after the breakthrough invention of the pipe '|'.

The pipe allowed the output of one program to be sent to the input of another. The tools philosophy was to have small programs to accomplish a particular task instead of trying to develop large monolithic programs to do a large number of tasks. To accomplish more complex tasks, tools would simply be connected together, using pipes.

All the core UNIX system tools were designed so that they could operate together. The original text-based editors (and even TeX and LaTeX) use ASCII (the American text encoding standard; an open standard) and you can use tools such as; *sed*, *awk*, *vi*, *grep*, *cat*, *more*, *tr* and various other text-based tools in conjunction with these editors.

Using this philosophy programmers avoided writing a program (within their larger program) that had already been written by someone else (this could be considered a form of code recycling). For example, command-line spell checkers are used by a number of different applications instead of having each application create its own own spell checker.

This philosophy lives on today in GNU/Linux and various other UNIX system-based operating systems (FreeBSD, NetBSD, OpenBSD, etc.).

Applications are almost anything running in userland that interact with end users. This excludes the kernel and everything tight to it like device drivers. Daemons, system services and utilities might be excluded or not depending on the context.

Here again, POSIX defines what are portable applications, i.e. programs written in ISO C and describe the various levels of conformance they are allowed to follow. Of course, non conforming applications can use whatever programming language and non portable facility they like.

In any case, applications use APIs (Application Programming Interfaces).

There are a number of Unix utilities that allow one to do such things as break text files into pieces, combine text files together, extract bits of information from them, rearrange them, and transform their content. Taken together, these Unix tools provide a powerful system for obtaining linguistic information.

GNU/GPL License:

The GNU General Public License (GNU GPL or GPL) is a widely used[6] free software license, which guarantees end users (individuals, organizations, companies) the freedoms to run, study, share (copy), and modify the software. Software that allows these rights is called free software and, if the software is copylefted, requires those rights to be retained. The GPL demands both. The license was originally written by Richard Stallman of the Free Software Foundation (FSF) for the GNU Project.

In other words, the GPL grants the recipients of a computer program the rights of the Free Software Definition[7] and uses copyleft to ensure the freedoms are preserved whenever the work is distributed, even when the work is changed or added to. The GPL is a copyleft license, which means that derived works can only be distributed under the same license terms. This is in distinction to permissive free software licenses, of which the BSD licenses and the MIT License are the standard examples. GPL was the first copyleft license for general use.

Prominent free software programs licensed under the GPL include the Linux kernel and the GNU Compiler Collection (GCC). Some other free software programs (MySQL is a prominent example) are dual-licensed under multiple licenses, often with one of the licenses being the GPL.

Free Software Foundation:

The Free Software Foundation (FSF) is a non-profit organization founded by Richard Stallman on 4 October 1985 to support the free software movement, which promotes the universal freedom to study, distribute, create, and modify computer software, with the organization's preference for software being distributed under copyleft ("share alike") terms, such as with its own GNU General Public License. The FSF was incorporated in Massachusetts, USA, where it is also based.

From its founding until the mid-1990s, FSF's funds were mostly used to employ software developers to write free software for the GNU Project. Since the mid-1990s, the FSF's employees and volunteers have mostly worked on legal and structural issues for the free software movement and the free software community.

Consistent with its goals, only free software is used on the FSF's computers.

About Multics :

Multics (Multiplexed Information and Computing Service) is a timesharing operating system begun in 1965 and used until 2000. The system was started as a joint project by MIT's Project MAC, Bell Telephone Laboratories, and General Electric Company's Large Computer Products Division. Professor Fernando J. Corbató of MIT led the project. Bell Labs withdrew from the development effort in 1969, and in 1970 GE sold its computer business to Honeywell, which offered Multics as a commercial product and sold dozens of systems.

Multics included

- a supervisor program that managed all hardware resources, using symmetric multiprocessing, multiprogramming, and paging
- an innovative segmented memory addressing system supported by hardware
- a tree structured file system
- device support for peripherals and terminals
- hundreds of command programs, including language compilers and tools
- hundreds of user-callable library routines
- operational and support tools
- user and system documentation

History and who started Linux

Linux is the first truly free Unix-like operating system. The underlying GNU Project was launched in 1983 by Richard Stallman originally to develop a Unix-compatible operating system called GNU, intended to be entirely free software. Many programs and utilities were contributed by developers around the world, and by 1991 most of the components of the system were ready. Still missing was the kernel.

Linus Torvalds invented Linux itself. In 1991, Torvalds was a student at the University of Helsinki in Finland where he had been using Minix, a non-free Unix-like system, and began writing his own kernel. He started by developing device drivers and hard-drive access, and by September had a basic design that he called Version 0.01. This kernel, which is called Linux, was afterwards combined with the GNU system to produce a complete free operating system.

Linux continued to be improved through the 1990's, and started to be used in large-scale applications like *web hosting*, networking, and database serving, proving ready for production use. Version 2.2, a major update to the Linux kernel, was officially released in January 1999. By the year 2000, most computer companies supported Linux in one way or another, recognizing a common standard that could finally reunify the fractured world of the Unix Wars. The next major release was V2.4 in January 2001, providing (among other improvements) compatibility with the upcoming generations of Intel's 64-bit Itanium computer processors.

Although Torvalds continued to function as the Linux kernel release manager, he avoided work at any of the many companies involved with Linux in order to avoid showing favoritism to any particular organization, and instead went to work for a company called Transmeta and helped develop mobile computing solutions, and made his home at the Open Source Development Labs (OSDL), which merged into The Linux Foundation.

Advantages of Linux:

FREEDOM!

Most Linux distros are free….. users do not need to pay for a copy, but this is only one aspect of freedom enjoyed by Linux users! In addition, Linux distros can be freely downloaded and legally installed on as many computers as you want and freely (and legally) given to other people. Because most distros are open source, you have access to the source code and can customize Linux to be whatever you want it to be; you can even create your own distro if you like!Linux is easy to install! In many instances, it is actually easier to install Linux to your computer than Windows.

LINUX IS VERY STABLE!

Linux systems rarely crash, and when they do, the whole system normally does not go down. The "blue screen of death" familiar to Windows users is not a worry for Linux users.

LINUX IS LESS VULNERABLE TO COMPUTER MALWARE!

Because most computer malware are designed to attack Windows (often throughActive X which is not typically found in Linux) the odds are considerably less for Linux to be infected with

a virus than Windows . The same holds true with spyware,trojans, and worms. While Linux malware does exist, they are relatively few in number and none have become widespread so far. While Linux is very secure by its nature, users should still employ good sense while surfing the Internet. As long as Linux users download and install only from their distro's official software repository, then security is greatly increased. One nice security feature In Linux is that files must be made to be executable by someone with administrator privileges, which requires a password. So even if a Linux virus is loaded on a Linux computer, it will not be able to run without the user who has administrator privileges intentionally making it executable. Another important aspect of Linux security is the fact that it is open source. Because the programming code is available for anyone to view, there are many eyes constantly examining it, which makes it highly difficult for malware to be hidden within the code. Also, security patches normally come much quicker to Linux than other operating systems because so many people are contributing to it.

LINUX TYPICALLY DOES NOT SLOW DOWN OVER TIME!

Unlike Windows, Linux does not easily become bogged down with spyware, viruses, trojans, etc., which can greatly reduce a computer's performance. Also, because Linux does not have a registry like Windows, it is not plagued with registry errors which can slow down a computer over time. Finally, the hard drives on Windows (especially Windows XP and older) computers need to be defragmented on a regular basis in order to maintain faster performance, due to being formatted in NTFS. On the other hand, because Linux is normally formatted in a different way using ext4 among others, there is no need to defragment a Linux hard drive.

LINUX CAN BREATHE NEW LIFE INTO OLD COMPUTERS!

If you have an older computer (especially Pentium III or later) laying around, you can install Linux and in essence have a new computer. In many cases Linux will run faster and you can do all of the basics such as browse the Internet, email, play games, and create and edit documents, spreadsheets, and PowerPoint presentations. It should also be mentioned that Linux runs great on newer computers as well.

WITH LINUX, YOU HAVE SO MANY CHOICES IN A WIDE VARIETY OF DISTROS!

Linux comes in all sizes and flavors, which offers a wide variety from which to choose the distro which will best suit your needs. Another advantage of this variety is the innovation that is taking place in the Linux world because it is open source.

WITH MANY LINUX DISTROS, YOU HAVE ACCESS TO FREE SOFTWARE WHICH NUMBERS IN THE THOUSANDS!

Popular distros such as Ubuntu, PCLinuxOS, andOpenSUSE offer excellent software repositories within their package managers where virtually any type of software can be downloaded and installed to your Linux system for free. This includes just about anything you can imagine, such as games, educational software, office suites, and much more! Some smaller distros, such as Peppermint OS, Lubuntu, Bodhi Linux, andPuppy Linux are based on Ubuntu and as a result have access to Ubuntu's software repositories. One very nice aspect of these repositories is that the software found in them has already been tested for compatibility and safety. For example, the thousands of free and open source software found in the Ubuntu Software Center has been tested and examined by Ubuntu, so a user can be confident that the software will be compatible with Ubuntu and will not include malware.

A SUPERIOR METHOD OF UPDATING SOFTWARE!

With Linux distros such as Ubuntu, OpenSUSE, PCLinuxOS, Fedora and many others, the majority of any software needed can be downloaded, installed, and updated from a central package management system provided by the distro. The result is a very smooth and seamless software updating process for Linux users.

Disadvantages of Linux:

MANY WINDOWS PROGRAMS WILL NOT RUN IN LINUX.

iTunes, Microsoft Office, Internet Explorer and many other Windows programs will not run natively in Linux. The good news is that there are decent ways around most of these problems. For example, music libraries can be managed with an iPod using programs such asAmarok, Banshee, or Rhythmbox in Linux. Mozilla Firefox and Google Chrome are

outstanding Internet browsers which can be used in the place of Internet Explorer. It is also possible to run iTunes in Linux using Wine, VirtualBox, or Parallels, though it is difficult to have good results. LibreOffice and OpenOffice are excellent office suites which can be used in the place of Microsoft Office, but while overall compatibility in both suites is good with Microsoft Office formats, it is not perfect.

THERE IS A SMALLER SELECTION OF PERIPHERAL HARDWARE DRIVERS FOR LINUX.

There is a smaller selection of peripheral hardware drivers (for printers, scanners, and other devices) in Linux as compared to Windows, though many new Linux hardware drivers are constantly being added. Closely related to this issue is the fact that not all Linux distros work with all sets of computer hardware, so a person may need to try more than one distro to find one which works well with his/her computer.

THERE IS A LEARNING CURVE FOR PEOPLE WHO ARE NEW TO LINUX.

Despite this, most Linux distros, especially the major ones, are very intuitive and user-friendly. Also, the desktop environments in Linux are in many ways similar to Windows in their appearance.

Linux distribution

A Linux distribution (often called a distro for short) is an operating system made from a software collection, which is based upon the Linux kernel and, often, a package management system. Linux users usually obtain their operating system by downloading one of the Linux distributions, which are available for a wide variety of systems ranging from embedded devices (for example, OpenWrt) and personal computers to powerful supercomputers (for example, Rocks Cluster Distribution).

A typical Linux distribution comprises a Linux kernel, GNU tools and libraries, additional software, documentation, a window system (the most common being the X Window System), a window manager, and a desktop environment. Most of the included software is free and open-source software made available both as compiled binaries and in source code form, allowing modifications to the original software. Usually, Linux distributions optionally include some proprietary software that may not be available in source code form, such as binary blobs required for some device drivers. Almost all Linux distributions are Unix-like; the most notable exception is Android, which does not include a command-line interface and programs made for typical Linux distributions.

A Linux distribution may also be described as a particular assortment of application and utility software (various GNU tools and libraries, for example), packaged together with the Linux kernel in such a way that its capabilities meet the needs of many users. The software is usually adapted to the distribution and then packaged into software packages by the distribution's maintainers. The software packages are available online in so-called repositories, which are storage locations usually distributed around the world. Beside glue components, such as the distribution installers (for example, Debian-Installer and Anaconda) or the package management systems, there are only very few packages that are originally written from the ground up by the maintainers of a Linux distribution.

Almost six hundred Linux distributions exist, with close to five hundred out of those in active development, constantly being revised and improved. Because of the huge availability of software, distributions have taken a wide variety of forms, including those suitable for use on desktops, servers, laptops, notebooks, mobile phones and tablets, as well as minimal environments typically for use in embedded systems. There are commercially backed distributions, such as Fedora (Red Hat), openSUSE (SUSE) andUbuntu (Canonical Ltd.), and entirely community-driven distributions, such as Debian, Slackware, Gentoo and Arch Linux. Most distributions come ready to use and pre-compiled for a specific instruction set, while some distributions (such as Gentoo) are distributed mostly in source code form and compiled locally during installation.

Linux kernel

The Linux kernel is a Unix-like computer operating system kernel. It is used world-wide: the Linux operating system is based on it and deployed on both traditional computer systems such as personal computers and servers, usually in the form of Linux distributions, and on various embedded devices such as routers and NAS appliances. The Android operating system fortablet computers, smartphones and smartwatches is also based atop the Linux kernel.

The Linux kernel was conceived and created in 1991, by Finnish computer science student Linus Torvalds for his personal computer and with no cross-platform intentions, but has since expanded to support a huge array of computer architectures, many more than other operating systems or kernels. Linux rapidly attracted developers and users who adapted code from other free software projects for use with the new operating system. The Linux kernel has received contributions from nearly 12,000 programmers from more than 1,200 companies, including some of the largest software and hardware vendors.

The Linux kernel API, the application programming interface (API) through which user programs interact with the kernel, is meant to be very stable and to not break userspace programs (some programs, such as those with GUIs, rely on other APIs as well). As part of the kernel's functionality, device drivers control the hardware; "mainlined" device drivers are also meant to be very stable. However, the interface between the kernel and loadable kernel modules (LKMs), unlike in many other kernels and operating systems, is not meant to be very stable by design.

The main differences

BSD is considered “university Unix”, or hobbyist Unix, because it came out of UC Berkeley in California

System V was considered more commercial

Sun OS was based on BSD, but Sun eventually moved to Solaris, which was System V based

SCO and HP-UX were also based on System V

One of the main differences was the location of binaries. System V standardized configurations, software installation, and handling network programming, which was in line with its corporate focus

System V placed its files in /usr/bin/ and /usr/sbin

BSD placed its files in /bin/ and /sbin/

Another big difference is in startup scripts: BSD used a script in /etc/rc to initialize itself and didn't use runlevels. The /etc/rc file is what files were run by init. To avoid having to edit /etc/rc, BSD variants supported a site-specific /etc/rc.local file that runs near the end of the boot process. Later BSD's, including FreeBSD and beyond, executes scripts out of the /etc/rc.d directory

System V uses what's now called SysV (Sis Vee) Style Init. SysV Style Init uses what are called runlevels, and a SysV system is always in exactly one runlevel. These include normal operation, single user mode, shutdown, and others. When you switch from one runlevel to another a series

of scripts are run before and after.

Notes

BSD Unix was developed at UC Berkeley.

System V is pronounced "System Five", and was developed by AT&T.

Over time, the two types have blended significantly, and modern operating systems (such as Linux) tend to have features of both.

There is significant consolidation in the Unix and Linux worlds. Expect to see more of this blending as this happens.

One big difference between BSD and Linux is that Linux is a kernel while BSD is an operating system. That’s the biggest difference between BSD and Linux: Linux is a a collection of little pieces, while BSD is one thing. - Read more at: http://scq.io/1Ctorwkj#gs.0ORtqCs