COURSE GUIDE

LINUX FOR IT MANAGERS COURSE GUIDE

Nicholas Kimolo

Through the support of the

COMMONWEALTH OF LEARNING

Copyright

Acknowledgements

I am grateful to the following for their assistance during the development of this course guide:

To my God who knows all. The everlasting foundation of knowledge and wisdom.

The Commonwealth of Learning (COL – www.col.org) for unrelenting support specifically mentioning Paul West - for believing in me, John Lesperance and Anthony Ming for review of material.

Ms. Nodumo Dhlamini and Dr. Jason Githeko from RUFORUM (www.ruforum.org) and Egerton University respectively for review and critique.

The Linux community and especially The Free and Open Source Software Foundation for Africa (FOSSFA – www.fossfa.net), the Free/Libre Open Source Software for Education (FLOSS4Edu – www.wikieducator.org/FLOSS4Edu), the WikiEducator community (www.wikieducator.org) – especially Prof. Wayne Macintosh, the founding father of WikiEducator for believing in Open Educational Resources and Linux , The LinuxIT , The Linux Professional Institute -(LPI) and The Linux Documentation Project-(TLDP) not forgetting the developers of the Creative Commons License (www.creativecommons.org) to which one of its licenses is used to copyleft this material.

To my dear wife – Martha Mutua - for guaranteed, unshaken support and understanding. To which this manual is dedicated to.

Contents

About this course GUIDEguide

Linux for IT Managers has been produced through the support of the Commonwealth of Learning and other individual and/or organizations involved in Free and Open Source Software. This Guide does not seek to promote any particular Linux vendor or distribution.

This course guide is structured as outlined below.

How this course GUIDEis structured

The course overview

The course overview gives you a general introduction to the course. Information contained in the course overview will help you determine:

- If the course is suitable for you.
- What you will already need to know.
- What you can expect from the course.
- How much time you will need to invest to complete the course.
- The overview also provides guidance on:
- Study skills.
- Where to get help.
- Course assignments and assessments.
- Activity icons.
- Modules.

We strongly recommend that you read the overview *carefully* before starting your study.

The course content

The course is broken down into modules. Each module comprises:

- An introduction to the module content.
- Module outcomes.
- New terminology.
- Core content of the module with a variety of learning activities.
- A module summary.
- Assignments and/or assessments, as applicable.

Resources

For those interested in learning more on this subject, we provide you with a list of additional resources at the end of this course GUIDE; these may be books, articles or web sites.

Your comments

After completing LINUX FOR IT MANAGERS COURSE GUIDEwe would appreciate it if you would take a few moments to give us your feedback on any aspect of this course. Your feedback might include comments on:

- Course content and structure.
- Course reading materials and resources.
- Course assignments.
- Course assessments.
- Course duration.
- Course support (assigned tutors, technical help, etc.)

Your constructive feedback will help us to improve and enhance this course.

Course overview

Welcome to Linux for IT Managers Course Guide

Computer Software has evolved exponentially in the past three (3) decades to be one of the key drivers of technology innovation and change in the field of Information and Communication Technology (ICT). In recent years, computer software has grown to be of wider importance than the hardware and the physical computing resources used to run that software. Computer Software has been known to contribute over 30% of the total cost of project implementation of common ICT projects. Computer Software makes it possible to use a piece of hardware or computing resource. With the trend towards the development of hardware platforms that can run multiple software platforms coupled with Virtualization and the delivery of Software and a Service, the need for appropriate software cannot be over emphasized.

Computer software can be grouped into two broad categories i.e Proprietary Software and Open Source Software. This categorization is broadly based on the availability of the programs source code controlled through an appropriate license or regulation and not necessarily on the disclosure of the source code that drives that particular application. The computer program source code can be disclosed and shown to all or a groups of persons but yet still remain as proprietary software. On the other hand, open source software is a type of computer software that focuses on the total disclosure of its source code and the subjection of the source code to an appropriate license that encourages the participation of a wider community in its enhancement and modification.

Though this manual is focused on Free and Open Source Software, of which the Linux Operating System is one of them, and its wide application areas, it is not the intention of the author(s) to exclude or underrate the availability of possible proprietary software alternatives. The decision on the implementation of a particular piece of software should be based on a sound business model that support's specific business requirements. In the consideration of which software to use for which need, it is important to maintain a holistic approach where other factors, not necessarily considered in this manual, should be addressed. In line with this way of thinking, this manual will seek to mention available proprietary software that can be used to achieve similar results.

This manual is organized into Nineteen (19) modules covering various topics as summarized below:

MODULE	MODULE TOPICS
1 – Introduction to Free and Open Source Software (FOSS) and Linux	In this Module you will be introduced to the various types and categorization of Computer Software, the philosophy behind Free and Open Source Software and the historical evolution of the Linux Operating System.
2 – Linux Installation and Configuration	You will be introduced to the: • Linux File System Hierarchy Standard (FHS) and Linux File Systems including but not limited to the Extended File Systems (ext2, ext3), Reiser and Journaling File Systems; • Linux Device Management including Partitioning Schemes and Disk formatting; • Linux Hardware Requirements and how to check compatibility with a particular Linux Distribution; and • Various types of Linux installations including installing Linux from boot disk, Installation CD/DVD ROMS and the Network. • Linux Installation Graphical User Interface (Windowed and Text)
3 - The Linux Command Line Structure 1	In this Module you will be introduced to the Linux Command Structure (Variables and command options), the Interactive Shell, metacharacters and quotes
4. Linux File Management	In this module you will learn how to: • Carry out Basic File Management Operations including creating. Deleting, Editing, Renaming, Copying etc file contents and folders i.e Moving around the file system; • Find files and directories; and Work with hard and soft links.
5 - Linux Post installation Activities	In this Module you will learn the activities that you can perform upon a basic linux installation. More specifically, you will learn how: • To configure a Linux Boot Manager and the Linux Boot process (From Boot Loader to Bash) ; and • To configure peripheral devices including printers and networking
6 - Devices and Linux File System Management	In this Module you will learn how to: • Create partitions and File Systems • Maintain the Integrity of the File System • File Permissions • Monitoring Disk Usage

MODULE	**MODULE TOPICS**
	• Control File System Mounting and Unmounting • Set and View Disk Quotas
7 – Process Management	In this Module you will learn about how to manage running linux processes including: • Managing the init process • Viewing running processes • Modifying various attributes of running processes • Manage runlevels and system initialization from the CLI and configuration files (for example: /etc/inittab and init command, /etc/rc.d, rc.local)
8 - Groups and User Management	Here you will learn how to: • Create new users • Work with groups • User and groups configuration files • Command line options • Modifying accounts and default settings
9 - Text Manipulation	In this module you will learn how to manipulate text in Linux. The learner will cover the following: • The vi Editor: Modes, Inserting,Deleting,Copying,Searching,Undoing and Saving • Regular Expressions • The grep family • The sed Stream Editor • Basic Shell Scripting - Customize and Use the Shell Environment
10 – The Linux Kernel	In this module the learner will learn about: • The Modular Linux Kernel • Routine Kernel Recompilation • Manage Kernel Modules at Runtime • Reconfigure , Build and Install a Custom Kernel and Modules
11 - Advanced Bash Scripting	In this module you will learn about: • The bash environment and bash scripting essentials • Logical Evaluations and Loops • Handling user input • Working with numbers

MODULE	MODULE TOPICS
12. Software Package Installation	You will learn how to install Linux packages from source and readymade packages including: • Debian Packages (.deb) and using apt-get utilility (command line and with synaptic) • Red Hat Package Manager (RPM)
13. Linux Windowing Environment	In this module you will learn about how to: • Install and Configure XFree86 • Set Up xdm • Identify and Terminate Runaway X Applications • Install and Customize a Window Manager Environment
14 – Linux System Administration	In this module you will learn about: • Logfiles and configuration files • Log Utilities • Automating Tasks • Backups and Compressions • Linux Help and Documentation • Managing the print service
15 - Linux Networking Services	This module will cover the implementation of Network configuration in linux including Network Interface Notation, Host configuration, Start and Stop Networking, Routing and Troubleshooting Network connections in Linux
16 - Setting up Basic Networking Services: DNS, DHCP and LDAP	Upon completion of this module you will: • Understand DNS, DHCP and Lightweight Directory Services(LDAP) Install and Configure DNS, DHCP and LDAP
17 - Setting up the Internet Gateway and Web Services	This module will enable you to understand Network Proxy Servers, Install and configure the Squid Proxy Server and: • Configure the Squid Proxy Server to offer controlled, authenticated and validated internet gateway access • Use SquidGuard to offer content filtering services • Installation and Configuration of APACHE Webserver • Advanced Apache Web server configuration • Apache Web server Performance tuning
18 – Email Gateway	This module will cover the following topics: • Operate and Perform Basic Configuration of Sendmail MTA’s

MODULE	**MODULE TOPICS**
	• Implementing Spam and Content Filtering • Configuring Virus Filtering with MTA’s • Deploying secure Webmail services
19 – Linux Network Security	This module will cover the following topics: • Perform Security Administration Tasks - Encryption • Set Up Host Security - Configure security environment files • Set Up User-Level Security • Minimization and Hardening of a Linux Server • Setting up a Stateful IPTable Firewall • Installing and Configuring of an Intrusion Detection System: Snort and Port Sentry • Security Tools: SSH, LSOF, NETSTAT,TCPDUMP and NMAP

Linux for IT Managers Course Guide—is this course for you?

This manual course content can be used by learners preparing for other certifications including Linux+ and the Linux Professional Institute (LPI) Certification. The main focus of this manual, though, will be to provide real life, hands-on solutions that commonly affect Technical and Mid Level IT Managers. This manual is intended for the following groups of professionals:

- Technical IT professionals (System Administrators, Network Administrators and Technical Specialists);
- IT Security Professionals;
- Database Administrators;
- Computer Programmers and Software Developers;
- IT Consultants; and
- Any other ICT professional with interest in Linux and Open Source Software.

This manual will assume that the learner has previous computer knowledge, but not necessarily in Linux. The learner should have prior knowledge on the use of computers at a basic level. Programmers, novice System Administrators, and new users of Linux looking for comprehensive Linux instructions will find this manual of benefit.

Course outcomes

Upon completion of this course guide you will be able to:

Outcomes

.1. Understand the philosophy of Free and Open Source Software (FOSS)

.2. Understand the fundamentals of the Linux Operating System and use appropriate Open Source Software to implement common applications

.3. Know about proprietary alternatives to all the Free and Open Source Software (FOSS) used in this guide

This manual will seek to achieve the following three (3) broad objectives:

1. Introduce the learner to Open Source Software in general and specifically, the Linux Operating System as applicable to Mid Level IT Managers i.e System, Network and IT Security Administrators;
2. Equip the Mid Level Manager with the necessary fundamental knowledge to effectively operate within a Linux based environment; and
3. Provide Mid Level IT Managers with the necessary knowledge and skills to set up and run common Linux based application services including Mail Gateways, Internet Gateways, Web Services, IT Security Services and other common application services for their organizations or institutions

Timeframe

This course is scheduled to take nineteen (19) days where you will cover a lesson or module every day.

Study skills

As an adult learner your approach to learning will be different to that from your school days: you will choose what you want to study, you will have professional and/or personal motivation for doing so and you will most likely be fitting your study activities around other professional or domestic **responsibilities.**

Essentially you will be taking control of your learning environment. As a consequence, you will need to consider performance issues related to time management, goal setting, stress management, etc. Perhaps you will also need to reacquaint yourself in areas such as essay planning, coping with exams and using the web as a learning resource.

Your most significant considerations will be *time* and *space* i.e. the time you dedicate to your learning and the environment in which you engage in that learning.

We recommend that you take time now—before starting your self-study—to familiarize yourself with these issues. There are a number of excellent resources on the web. A few suggested links are:

http://www.how-to-study.com/

The "How to study" web site is dedicated to study skills resources. You will find links to study preparation (a list of nine essentials for a good study place), taking notes, strategies for reading text books, using reference sources, test anxiety.

http://www.ucc.vt.edu/stdysk/stdyhlp.html

This is the web site of the Virginia Tech, Division of Student Affairs. You will find links to time scheduling (including a "where does time go?" link), a study skill checklist, basic concentration techniques, control of the study environment, note taking, how to read essays for analysis, memory skills ("remembering").

http://www.howtostudy.org/resources.php

Another "How to study" web site with useful links to time management, efficient reading, questioning/listening/observing skills, getting the most out of doing ("hands-on" learning), memory building, tips for staying motivated, developing a learning plan.

The above links are our suggestions to start you on your way. At the time of writing these web links were active. If you want to look for more go to www.google.com and type "self-study basics", "self-study tips", "self-study skills" or similar.

Need help?

Help

In case of any need for assistance or enquiry you can join this guide's mailing list found at http://www.colwiki.org/Linux_for_IT_Managers_Training_Manual . You may also join the google group "Linux for IT Managers":

Homepage - http://groups.google.com/group/linux-for-it-managers?hl=en

Group email- linux-for-it-managers@googlegroups.com.

You may also contact the author directly by sending an email to: Nicholas at futuristic dot co dot ke . You may also call on +254720349420.

Assignments

Assignments

At the end of every module, you shall be presented with a series of questions that you can answer as part of your assignments.

Getting around this course GUIDE

Margin icons

While working through this Course Guide you will notice the frequent use of margin icons. These icons serve to “signpost” a particular piece of text, a new task or change in activity; they have been included to help you to find your way around this course guide.

A complete icon set is shown below. We suggest that you familiarize yourself with the icons and their meaning before starting your study.

Activity	**Assessment**	**Assignment**	**Case study**
Discussion	**Group activity**	**Help**	**Note it!**
Outcomes	**Reading**	**Reflection**	**Study skills**
Summary	**Terminology**	**Time**	**Tip**

Module 1

course content can be used by learners preparing for other certifications including Linux+

Introduction to Free and Open Source Software (FOSS)

Introduction

This module will focus on Free and Open Source Software as generally defined in the table above. Upon completion of this module you will be able to:

Outcomes

- Understand the various types and categorization of Computer Software; and
- Understand the Historical Evolution of Free and Open Source Software and the difference between Free Software and Open Source Software.

Definitions and Historical development of Open Source Software

Free and Open Source Software (FOSS) can be available commercially or as non-commercial software. The "free" in open source software refers to the Free Software Foundation's (www.fsf.org) definition of 'What is Free Software'. The freedoms at the core of free software are defined as:

1. The freedom to run the software for any purpose;
2. The freedom to study how the software works and adapt it to your needs;
3. The freedom to redistribute copies so you can help others; and
4. The freedom to improve the software and release your improvements to the public, so that everyone benefits.

Originally hardware was the major revenue stream for most of the Computer Vendors. From the early 1960s to the early 1980s, revenues in computer business were generated through selling and supporting hardware. For every hardware device, a special operating system was developed and deployed.

The users of these systems were highly specialised IT experts. They were the ones primarily responsible for the development of additional software.

Many efforts were dedicated to build an operating system that could be deployed on multiple hardware platforms. The most prominent example was Unix, which developed at the AT&T Laboratories and was published in 1969. Commercial users had to pay high license fees for using Unix, whereas academic institutions could use the software for a nominal charge. Consequently, Unix was the basis for the development of the Internet technologies. Many of these technologies were developed at universities and computer companies research laboratories, where Unix was deployed. Sharing the source code among software developers was commonplace. This tendency was reinforced by the emergence of computer networks like the Usenet that was started in 1979 to link the Unix community.

A critical event in the early 1980s for cooperative software development was the turnaround in AT&T's licensing policy. Unix became restricted to those who paid for the license to use is. Following this first step into the direction of closed source, the hardware companies IBM, HP and DEC started to develop proprietary Unix operating systems. They imposed "non-disclosure agreements" on the programmers dealing with the software and recruited many developers for commercial software development who had formerly contributed to cooperative and shared software development.

At that time, the programmer Richard Stallman worked in software development at the MIT. In 1984, he started a project to develop a free alternative of the Unix operating system. In addition, he established a special license, the GNU (named for Gnu'sNot Unix) license, which was supposed to ensure that the software is indeed free and open for everyone. In order to support the GNU project, Stallman founded the Free Software Foundation (FSF) in 1985. Although linked often to the Open Source movement, Stallman is a proponent of Free Software, which goes much further in its demands (See http://www.gnu.org/philosophy/free-sw.html). Nevertheless, the GNU General Public License (GPL, see "Licenses") is central to the evolution of the Open Source phenomenon and has been used in many important projects.
In the GPL, the principle of "Copyleft" is realised: It means that every copy of a program governed by the GPL, even if modified, must be subject to the GPL again.

The FSF's philosophy behind software development provided great motivation for the Free Software community. But it also resulted in antipathy from many businesses which partly remains until today. The most prominent debate over the implications of Open Source Software, especially the GPL, and its effects on innovation takes place between Microsoft and Free/Open Source Software advocates, although such discussions are commonplace in more prosaic settings as well.

In the early 1990s, along with the increasing use of the Internet and the success of the World Wide Web, many new Open Source projects emerged. The most prominent example is Linux. Linux is a Unix-like operating system targeted to run on a personal computer. It was developed by the Finnish computer science student Linus Torvalds who used the GNU software tools. In 1991, he released the code of an experimental version under the GPL to a newsgroup and asked for comments and improvements.

Within the last decade, Linux developed into a powerful operating system. The project showed characteristics that are typical for successful Open Source Software development over the Internet. Eric Raymond, another central OSS developer and advocate, describes OSS development coordination as "Bazaar style," opposed to the "Cathedral" approach taken in classical software development, where development is organised in a more hierarchic, top-down and planned way. Linux has a modular structure, so individuals or groups of developers can focus on one part of the program. The principle of "Release often, release early" in combination with a constant peer-reviewing process ("Given a thousand eyes all bugs are shallow") is also opposed to commercial software development.

Linux was used increasingly in combination with the GNU tools. Because the operating system is central to IT infrastructure, it eventually became relevant for business use. In 1997, the Open Source Initiative (OSI) was founded in order to establish a more pragmatic approach to software licensing. The OSI was based on the "Debian Free Software Guidelines," which had been published in 1995. The central people for this development were Eric Raymond and Bruce Perens. Their aim was to promote OSS in commercial use because they believed that both the Free/Open Source community and the business world could benefit from wider OSS dissemination.

The OSI developed the Open Source Definition (OSD). The definition (See http://www.opensource.org/docs/definition.php) is not a license itself, but a guideline and trademark for OSS software licenses other than the GPL. Licenses according to the OSD guarantee several freedoms to software users, including commercial users. The "viral" effect of the GPL is not a requirement for OSD-approved licenses. In order to raise acceptance of OSS in the business world, the term Open Source Software instead of Free Software was established and widely accepted.

The 1990s experienced a significant rise in attention paid to Open Source projects. Many companies from the IT industry began to support the projects. IBM, for example, supports a variety of Open Source projects. In 1998, Netscape was the first prominent company to release a proprietary software product as Open Source software.

This trend has continued to grow with the rise of successful linux distributions for the desktop. Both Open Source and Free Software continue to enjoy growing success and wide recognition. While some refer to free and open source as competing movements with different ends, the author of this guide does not see free and open source software as either distinct or incompatible. According to Wired Open Source Census Research (http://www.marketwire.com/press-release/Openlogic-886791.html):

1. Ubuntu (45%) and Debian (14%) are the most used Linux distributions among participants with Linux machines
2. More than half of the open source software found has been on Windows machines

3. The number of unique installed open source packages ranged from 22-62 per machine

These trends have been noted out by other researchers. This could be an indicator of the varied number of people who use Free and Open Source Software. Looking into the crystal ball to forecast the future is difficult on many domains, but specifically in the fast moving Open Source domain. It is expected that continuous consolidation and convergence of multiple technologies will happen with the commercial software vendor creating room for successful open source vendors. Open Source adoption in the enterprise will continue, in the application infrastructure space the use of Open Source is already common sense, but more and more Open Source solutions will be viable candidates also for typical business solution domains. Web 2.0 will continue and shall accelerate the adoption of Open Source in the Enterprise even more. More commercially available products will be based on Open Source software. New open standards such as Open Social or Google Android will be the base of many new Open Source project and initiatives.

Types of Computer Software Forms

Software can be defined as of four different forms as summarized in the table below:

	Source Code Open - Yes	**Source Code Open -No**
Price for the User – Non Grantis (> 0)	Non Commercial OSS	Freeware/Shareware
Price for the user Gratis (Price = 0)	Commercial OSS	Proprietary/Commercial Software

A brief description of the various forms of Software is provided below:

1. **The classical proprietary/commercial software:** This is software that is typically distributed in binary form only. The source code is not available.

2. **Shareware:** Software that is typically free for an initial period, but generally would require a license to be bought after testing. The source code is not available. Freeware on the other hand has no license fee at all, at least not for the freeware but maybe for a complementary product. The source code is not available.

3. **Non Commercial Open Source Software:** No license fee at all. The source code of this software is available. Users can use the Software & Freely Redistribute it.

4. **Commercial Open Source:** This is software that one is required to pay for access to the source code.

Reflection

See the Microsoft Windows Historical Development process here: http://en.wikipedia.org/wiki/History_of_Microsoft_Windows . Think about what could have caused the earlier success of Microsoft Windows as an Operating System compared to the Linux Operating System.

Submit your thoughts and ideas to this course guide website indicated in the "About this Course Guide Section".

Module summary

Summary

In this module you learned about the historical evolution of Free and Open Source Software. Though there is a significant difference between Free and Open Source Software philosophically, this has no effect of the compatibility or applicability of the software developed.

You also learned that Free and Open Source Software can be available as commercial software. That "free" does not mean zero cost but in essence mean the freedom to apply certain rights granted by the software.

In the next lesson you will learn how to install and configure a basic linux installation

Assignment

Assignment

For your assignment in this module:

Visit http://www.opensource.org/docs/definition.php and http://www.gnu.org/philosophy/free-sw.html. Study the various foundation documents for both Open Source Software and Free Software. Note the differences and indicate the rationale of the difference and whether they are significant to you.

Module 2

Linux Installation and Configuration

Introduction

Linux comes in many forms including in CDROMs, DVDs, iso Images etc. Whichever form your Linux is contained in, you should be able to install it in your computer and use it. Many of the modern Linux distributions nowadays come with easy to follow user interfaces that you can run through and install.

This manual will not focus on specific installation procedure for any distribution but will highlight the building blocks to behind the scene installation activities.

The process of installing Linux involves copying files into the installation media. System files are more than just names for a collection of bits and bytes. Files are the central concept behind a variety of functions. Files not only store information but also allow applications to communicate, provide access to hardware devices, represent folders of other files, act as pointers to information, or (virtually) connect machines over a network. Files are the central concept behind Linux and Unix-like systems. Almost everything is treated as a file: all directories, pipes to other processes, the interface to hardware devices, even pointers to files (links). There are even virtual files that give a user access to kernel structures.

Whereas a file refers to a single entity, a file system describes the way files are stored on media. The term media includes the obvious hard or floppy disk, USB Disks and CD-ROMs, as well as a network service or the RAM of your machine. Each file system type implements different properties. Not every kind of file can reside on any file system, and not every file system type supports every medium. Files have properties that determine the file type. A common property of all files is that they have a set of permissions, which is the designation that indicates which group it belongs to and who is the owner. Before we discuss file systems or permissions and ownerships, let's take a closer look at the types of files.

- **Regular Files:** Regular files are used to store data on a file system. This is the "common sense" type of file, a container for persistent data.
- **Directories:** Directories hold other files. Because modern file systems are organized in hierarchies, you need something to hold the different levels of the hierarchy. That is a directory's purpose. If

you're not used to referring to directories as files, remembering that they are nothing other than files containing a list of other files is important.

- **Special Device Files:** Special device files are simply interfaces to a device. Two kinds exist: buffered device files and unbuffered device files. The buffered special device files are called **block device** files; the unbuffered ones are the **character device** files.
- **Regular Pipes:** A pipe is a connection between two processes. It's treated like a file inside the application, yet doesn't have a representation in the file system. This type of file is interesting only if you mean to write an application. From a user level, pipes are an easy method to use to concatenate commands and give the output from one application as input to another one.
- **Named Pipes:** Named pipes are like pipes but are represented in the file system. They are also used for interprocess communication, but they can exist without any process accessing them.
- **Sockets:** Sockets are similar to pipes and perform the same functions. The difference is that sockets are used to communicate over a network. The details are important only if you want to use sockets in your application. Further details on sockets are beyond the scope of this book. You can find excellent sources that explain the concept in greater depth.
- **Hard Links:** A hard link is a sequential entry in a directory structure for an existing file. It's like a new name for a file.
- **Soft Links:** A soft link is a pointer to a file.

Executables are not a separate file type and are therefore not included in this list. In Linux systems, being executable or not is a property determined by the file permissions and does not depend on the type of file. Remember that some files can be executed only by designated users or user groups.

With this understanding, you will learn the following in this lesson:

Outcomes

- Linux File System Hierarchy Standard (FHS) and Linux File Systems including but not limited to the Extended File Systems (ext2, ext3), Reiser and Journaling File Systems;
- Linux Device Management including Partitioning Schemes and Disk formatting;
- Linux Hardware Requirements and how to check compatibility with a particular Linux Distribution; and
- Various types of Linux installations including installing Linux from boot disk, Installation CD/DVD ROMS and the Network.
- Linux Installation Graphical User Interface (Windowed and Text)

The Linux File System

At the core of any Linux operating system is the Kernel. The Linux Kernel is the core of the operating system and provides the ability for software to access the hardware systems. This Kernel and any of the other applications or systems running on a Linux system work through files. Almost all of the ways an operating system interacts with its users, applications, and security model are dependent upon the way it stores its files on a storage device. It is crucial for a variety of reasons that users, as well as programs, be able to refer to a common guideline to know where to read and write files.

The table below summarizes a logical arrangement of files within the Linux Operating System:

	Shareable This are files that are not available to other hosts	**Unshareable** Accessible by various hosts
Variable This are files that can change at anytime without intervention	/var/mail /var/spool/news /home	/var/run /var/lock /tmp
Static Do not change without an intervention from the system administrator	/usr /opt	/etc /boot

The reason for looking at files in this manner is to help correlate the function of the file with the permissions assigned to the directories which hold them. The way in which the operating system and its users interact with a given file determines the directory in which it is placed, whether that directory is mounted read-only or read-write, and the level of access each user has to that file. The top level of this organization is crucial, as the access to the underlying directories can be restricted or security problems may manifest themselves if the top level is left disorganized or without a widely used structure.

In order to achieve this, many Linux systems abide to the File System Hierarchy Standard (FHS) - See http://www.pathname.com/fhs/ . The FHS is a standard that define the names and locations of many files and directories. It is very important that a Linux system complies with this standard as it ensures compatibility with other compliant systems. The following defines the File Organization within a Linux File System.

The /root/ Directory

The root directory is the most important part of the whole system. During the boot process, this file system is mounted first and has to hold everything to bring the system to life. This requires certain things to be on this file system:

- Start-up data and utilities to set up the system configuration.
- All utilities needed to mount other file systems. In a networked environment, this may include the network utilities, as mounts may be made over NFS.
- Tools to repair broken file systems in order to perform a recovery after a system crash. In particular, this means utilities to restore backups from tapes, floppy disks, and other media.

On the other hand, the administrator's goal is to keep the root file system as small as possible. The smaller a file system is, the less likely it is to be corrupted in case of a system crash. The root file system holds essential utilities as well as major system configuration data in /etc, which means that it's not shareable. A small root file system means less required hard disk space in environments where as much data as possible is shared over a local network.

The /boot Directory

/boot contains everything that is required for the first stage of the boot process. The /boot directory stores data that is needed before the kernel begins executing user mode programs. It also holds backup copies of the boot records, sector map files, and other items that are not edited manually. Programs that access this data (mainly lilo) are located in /sbin, and their configuration files are in /etc.

The /dev/ Directory

The /dev/ directory contains file system entries which represent devices that are attached to thesystem. These files are essential for the system to function properly.

The /etc/ Directory

The /etc/ directory is reserved for configuration files that are local to the machine. No binaries are to be put in /etc/. The X11/ and skel/ directories are subdirectories of the /etc/ directory and are used to configure the X Windows environment which is the graphical windowing system of Linux

The /lib/ Directory

The /lib/ directory contains only those libraries that are needed to execute the binaries in /bin/ and /sbin/. These shared library images are particularly important for booting the system and executing commands within the root file system.

The /mnt/ Directory

The /mnt/ directory is for temporarily mounted file systems, such as CD-ROMs and floppy disks.

The /opt/ Directory

The /opt/ directory provides storage for large, static application software packages. A package placing files in the /opt/ directory creates a directory bearing the same name as the package. This directory in turn holds files that otherwise would be scattered throughout the file system, giving the system administrator an easy way to determine the role of each file within a particular package.

The /proc/ Directory

The /proc/ directory contains special files that either extract information from or send information to the kernel. Due to the great variety of data available within /proc/ and the many ways this directory can be used to communicate with the kernel.

The /sbin/ Directory

The /sbin/ directory is for executables used only by the root user or system administrator. The executables in /sbin/ are only used to boot and mount /usr/ and perform system recovery operations.

The /usr/ Directory

The /usr/ directory is for files that can be shared across a whole site. The /usr/ directory usually has its own partition, and it should be mountable read-only.

The /var/ Directory

Since the FHS requires Linux to mount /usr/ read-only, any programs that write log files or need spool/ or lock/ directories should write them to the /var/ directory.

A review of the Linux File System

Now that we've reviewed the basics, single files, it's time to take this review to another level -- file systems. We've used the term file system quite frequently up to now without a working definition. Because file system has two different meanings, it can become quite confusing. When we talk about file systems, **we usually think of directory trees. This is a hierarchical structure of directories that contain other directories and/or any kind of file.**

The second meaning of file systems **is the lower-level format, used on a given media to store files.**

Alternatives to File Systems Storing data to media can be done in many ways. The simplest one is to dump the raw data to the medium. This works fine but makes accessing the data inconvenient. You lack meta information such as size, type, or identifier. Matters are complicated if you want to store a second set of data to this medium. You have to remember the offset (where the first set ends), or you risk overwriting, which means losing data.This meta information can be supplied by special applications or made transparent for the user by a file system.

File systems organize files into logical hierarchical structures with directories, links, and so on. The Linux system supports over twenty-one different file system types: minix, ext2, iso9660, msdos, umsdos, vfat, proc, nfs, smb, ncp, hpfs, sysv, adfs, amiga-fs, ROM-fs,NTFS, joilet, Apple Macintosh fs, QNX fs, Coda, and ufs. A file system organizes the data on a block device. This can be a hard disk, a floppy disk, RAM, a flash RAM cartridge, or even a network link to a remote system. The file system itself doesn't know anything about the medium it uses. This is handled by the device driver. It's the job of the device driver to translate the address of a specific block to a physical position on a hard drive, a memory region, and so on.

The de facto standard file system on Linux systems is Extended 2 File System (ext2). The first file system for Linux was the Minix file system (Minixfs). It had many disadvantages. Partitions were limited to 64MB, filenames to 14 characters. The extended file system, introduced to the Linux world in April 1992, was mostly based on the Minixfs; it removed the previously mentioned limits, but still was far from perfect. This led to the development of the ext2 file system. It added better space allocation management to extfs, exhibited better performance, allowed the use of special flags for file management, and was extensible. It's been available since 1993 and is the most successful Linux file system to date.

When the extfs was added to the Linux kernel, a new layer entered the system, the so-called Virtual File System (VFS) layer. With this layer, the file system and the kernel were no longer one module. A well-defined interface between kernel and file system was created. VFS allows Linux to support many different file systems. All file systems appear identical to the kernel and to the programs running on the system. The VFS layer allows you to transparently mount different file systems at a time.

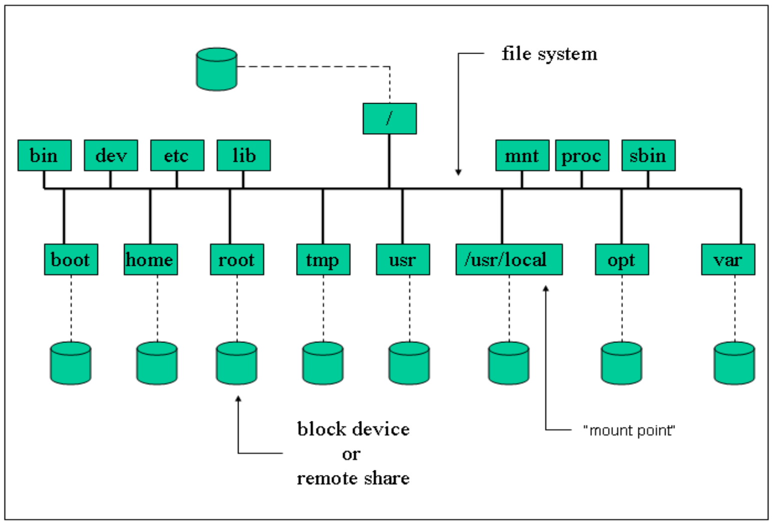
The ext2 file system is built on the premise that data is held in blocks of equal size on the storage medium. The block size can vary on different ext2 file systems, but within one file system, all blocks have the same length. The length is set when the file system is created by mke2fs. This strategy's disadvantage is that on average, half the block size is wasted for each file. Assuming that the block size is 1024 bytes, a file with 1025 bytes will use two blocks, just as a file with 2047 will. By handling the files this way, you reduce the CPU's workload.

Not all blocks on the file system hold actual file data. Some are used to hold information about the structure of the file system. Ext2 uses inode data structures for each file stored in the file system. An inode describes which blocks contain the data of the specified file, its type, owner, and group, the permissions and the date of the last access, the last modification, and its creation date. The inodes are held in inode tables. A directory is simply a file that contains pointers to the inodes of the files within this directory.

A detailed description of the whole ext2 file system can be found in David A Rusling's book The Linux Kernel, in Chapter 9, "The File System," available online at http://sunsite.unc.edu/LDP/LDP/tlk/tlk.html .

Partitioning Schemes

The figure below shows a possible partitioning scheme. The File System layout is a tree of directories and subdirectories. The physical resources with the data are **mounted** at specific locations on the file system called **mount points**. The root of the tree structure is called **root** and is represented by a forward slash "/". At boot time, the boot loader is told which device to mount at root. The leaves in this tree structure are subdirectories. During installation you will partition the hard drive and assign a size and a mount point for each partition.



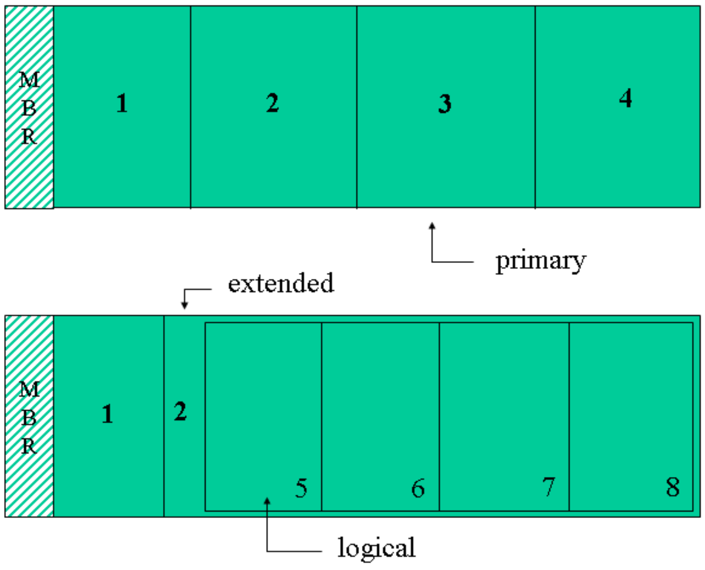
On a running Linux system, disks are represented by entries in the **/dev** directory. The kernel communicates with devices using a unique major/minor pair combination. All major numbers are listed in **/proc/devices**. For example the first IDE controller's major number is **3**: 1 – ramdisk, 2 – fd and 3- ideo . Hard disk descriptors in **/dev** begin with *hd* (IDE) or *sd* (SCSI), a SCSI tape would be *st*, and so on. Since a system can have more than one block device, an additional letter is added to the descriptor to indicate which device is considered.

	Physical block devices
hda	Primary Master
hdb	Primary Slave
hdc	Secondary Master
hdd	Secondary Slave
sda	First SCSI disk
sdb	Second SCSI disk

Disks can further be partitioned. To keep track of the partitions a number is added at the end of each physical device.

	Partitions
hda1	First partition on first hard disk
hda2	Second partition on first hard disk
sdc3	Third partition on third SCSI disk

IDE type disks allow 4 *primary* partitions, one of which can be *extended*. The extended partition can further be divided into *logical* partitions. There can be a maximum of 64 partitions on an IDE disk and 16 on a SCSI disk. The primary partitions (1,2,3,4) and (1,2,5,6,7,8).



During the installation you will be required to create various partitions to hold your file system. A linux system will require at least two partitions:

- **The root partition:** This is the partition where / (the root directory) will be located on a UNIX or UNIX-compatible system. This partition should not be confused with the /root directory which is the home directory of super user
- **A swap partition:** This is space on your hard drive that can be used as virtual memory. Virtual memory allows your computer to run large programs and perform complex tasks even if it does not have enough physical RAM to do the job. The amount of swap space required is subjective. To keep it simple, create one swap partition of either 64MB or 128MB (you may have to use 127MB instead depending on the program you use to create the partition). If you are short on RAM with plenty of drive space, go large. If you're short on drive space and have 128MB or more of RAM, go small.

Linux Installation

Because of the Open Source nature of the Linux Kernel, which allows anyone to modify or enhance the base kernel with other software, Linux in available in a wide variety of distributions. A linux distribution is a collection of software packages, utilities, and tools and is based on the version of a Linux kernel. Some distributions can be created with a specific purpose in mind e.g to set-up a firewall, or a security linux distribution (e.g backtrack),or for educational purposes etc but there are also general-purpose distributions, meaning that they come with a variety of software packages, tools and applications that can be installed by the end user resulting in flexibility.

Each distribution comes with its own characteristics and special tools, created by the distribution company to increase the value of their version of Linux. Most of these distributions are obtained for free (in keeping with the GPL License) but many companies also sell commercial distributions. Most Linux vendors center their business on a service support model for their distribution.

Often mention will be made for distinct changes with other key Linux distributions. It is also important to note that this manual does not seek to promote any particular distribution but will seek to expound on implementations that cut across any Linux distribution. This implies that the concepts described in this manual cut across majority of the Linux distributions out there. For a comprehensive listing of Linux distributions see http://www.distrowatch.com/

Note it!

There are many Linux distributions out there, most of them with very elaborate installation instructions. This manual shall not replicate those installation instructions here but shall instead point you to the relevant installation instructions for your module. The following are some of the key linux distributions and where you can find their download and installation instructions.

Distribution Name	*Download Site*	*Installation Instructions Site*
SuSE Linux Enterprise Server	http://download.novell.com/Download?buildid=xWohTS2zkSs~	*Pg 6-237 SuSE Linuxx Enterprise Server-Installation & Administration*
Ubuntu	*http://www.ubuntu.com/getubuntu/download*	*https://help.ubuntu.com/community/Installation/*
Fedora	*http://fedoraproject.org/get-fedora*	*http://docs.fedoraproject.org/install-guide/f11/*
CentOS	*http://mirror.centos.org/centos/5/isos/*	*http://www.centos.org/docs/5/*

Reflection

Having Installed Microsoft Windows Operating Systems before. Compare and contrast the installation process between your selected Linux Distribution and your Windows Installation Process.

Submit your comments and observations on your Wiki Page inside the http://www.colwiki.org/Reflections_and_Assignments webpage

Module summary

Summary

In this module you learned about the various Linux File types. You have also looked at the Linux File System Standard and learned how to install the SuSE Linux Enterprise Server 10 on your machine,

Assignment

Assignment

Attempt to access and download two different distributions above and attempt to install them on your computer. Note the differences between the distrubtion's installation processes.

Module 3

The Linux Command line Structure

Introduction

A basic way to interact with a computer system is to use the command line. The shell interprets the instructions typed in at the keyboard. That is the shell is the interface between a human user and the Linux Operating System. The shell prompt (ending with $ or # for user root) indicates that it is ready for user input.

You can access the Linux Shell by Pressing Ctrl+Alt+F1 or F2 or F3 or F4 or F5 or F6. These commands open up a shell terminal where you can be able to enter any Linux command. You may also access the terminal on the windows interface (Gnome Terminal on SLES – Click on **Computer** then Choose **"More Application"** then choose **Gnome Terminal**).This terminal runs a shell which is a programming environment which can be used to perform automated tasks. Shell programs are called scripts. The most common shells are summarized in the table below:

Most Common shells	
The Bourne shell	/bin/sh
The Bourne again shell	/bin/bash
The Korn shell	/bin/ksh
The C shell	/bin/csh
Tom's C shell	/bin/tcsh

Since the bash shell is one of the most widely used shells we shall concentrate on the use of this shell. This Module shall focus on the use of various Linux commands to perform basic file management commands in the Linux System. In this Module you will be:

Introduced to the Linux Command Structure (Variables and command options), the Interactive Shell, metacharacters and quotes plus perform the following:

- Carry out Basic File Management Operations including creating,Deleting, Editing, Renaming, Copying etc file contents and

Outcomes

folders

- Moving around the file system;
- Find files and directories; and.
- Work with hard and soft links.

Terminology	**Shell:**	A Unix shell is a command-line interpreter (see shell) and script host 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 command input as text for a command line interpreter to execute or by creating text scripts of one or more such commands.

The Interactive Shell

Shell commands are often of the form

command [options] {arguments}.

To print text to the screen the bash shell will use the **echo**

$echo “this is a short line”

The shell interprets the first “word” of any string given on the command line as a command. If the string is a full or relative path to an executable then the executable is started. If the first word has no “/” characters, then the shell will scan directories defined in the PATH variable and attempt to run the first command matching the string.

For example if the PATH variable only contains the directories /bin and /usr/bin then a command stored in /etc will not be found therefore, the full path needs to be specifies.

An alternative to typing the full path to an executable is to use a relative path. For example, if the user is in the directory where a particular program is stored then one can type:

```
$ ./<Program Name>
```

Variables

Shell variables are similar to variables used in any computing language. Variable names are limited to alphanumeric characters. For example

CREDIT=300 simply assigns the value 300 to the variable named CREDIT.

Initialize a variable:	Variable-Name=value (no spaces!!)
reference a variable:	$Variable-Name

```
CREDIT=300
echo $CREDIT
```

Export, Set and Env:

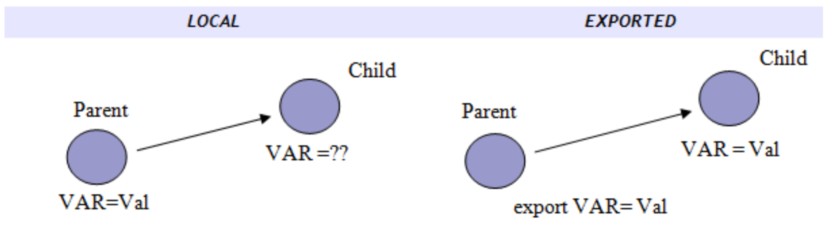
There are two types of variable: local and exported. Local variables will be accessible only to the current shell. On the other hand, exported variables are accessible by both the shell and any child process started from that shell.

The commands **set** and **env** are used to list defined variables

The set and env commands

Set	Lists all variables
Env	Lists all exported variables

A global variable is global in the sense that any child process can reference it.



Example: Make the CREDIT variable a global variable. Test whether it's listed with **set** or **env**.

```
export CREDIT

env | grep CREDIT
```

Start a new shell (child process) and verify that CREDIT is accessible. Can one start any shell and be sure that CREDIT is still declared?

List of common predefined variables

PREDEFINED VARIABLES	**MEANING**
DISPLAY	Used by X to identify where to run a client application
HISTFILE	Path to the users .bash_history file
HOME	The path to the user's home
LOGNAME	The name used by the user to log in
PATH	List of directories searched by the shell for programs to be executed when a command is entered without a path.
PWD	The current working directory
SHELL	The shell used (bash in most Linux distributions)
TERM	The current terminal emulation

Special variables

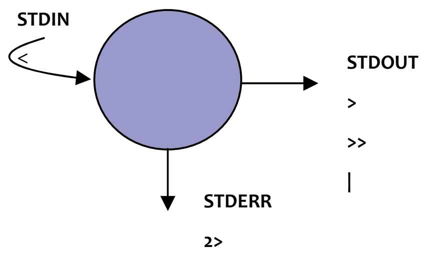
The next few variables are related to process management.

$!	represents the PID value of the last child process
$$	represents the PID of the running shell
$?	is 0 if the last command was executed successfully and 1 otherwise

Input Output Redirection

Linux processes normally open three standard file descriptors which enable it to process input and output. These standard descriptors can be redefined for any given process. In most cases the **stdin** descriptor is the keyboard, and the two output descriptors, **stdout** and **stderr**, is the screen.

A process and it's 3 descriptors



Numerical values for stdin, stderr and stdout

stdin	0
stdout	1
stderr	2

STDOUT REDIRECTION

program > file

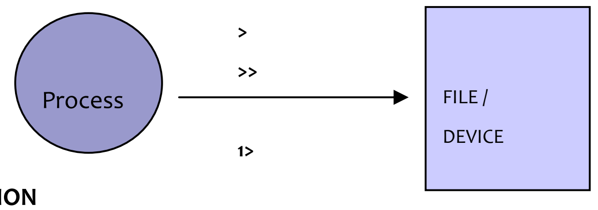
The data flows from left to right for example

```
fdisk –l > partions.txt
```

This will run the **fdisk** utility and output the result to the *partitions.txt* file. No output is visible. Also notice that the shell will read this line from the right. As a result, the *partitions.txt* file will be created first if it doesn't exist and overwritten if the '>' operator is used.

The '>>' operator will append output to a file.

STDOUT Redirection



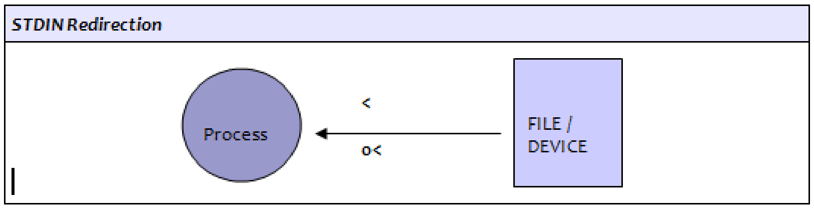
STDIN REDIRECTION

program < file

In this case data flows from right to left. The '<' operator is only used for **stdin** and cannot be used for **stdout**.

If the file *instructions* contains on each line the letters *p*, *m*, and *q* then the next example would cause **fdisk** to print the partition table of */dev/hda*, print the utility's help screen and finally quit:

```
fdisk /dev/hda  < instructions
```

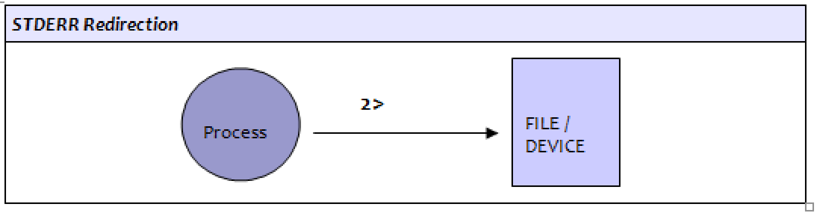


STDERR REDIRECTION

program 2> errorfile

stdin, stdout and stderr are represented by 0, 1 and 2 respectively. This allows one to select the **stderr** stream:

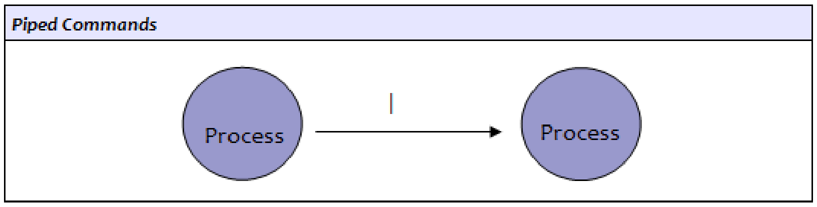
```
find / 2> /dev/null
```



PIPED COMMANDS

program1 | program2

Pipes are represented by the “|” symbol. The data stream goes from the left to the right. The next figure illustrates how the **stdout** for one process is redirected to the **stdin** for another process.



```
cat /var/log/messages | less
```

NB Multiple output redirects are parsed from right to left, so the following commands are not equivalent.

```
Do-command  2>&1  >logfile
Do-command  >logfile  2>&1
```

Metacharacters and Quotes

Metacharacters are characters that have special meaning for the shell. They are mainly used for *file globbing* that is to match several files or directory names using a minimum of letters.

The input (<), output (>) and pipe (|) characters are also special characters as well as the dollar ($) sign used for variables. We will not list them here but note that these characters are seldom used to name regular files.

Wildcards

- The * wildcard can replace any number of characters.

ls /usr/bin/b*	lists all programs starting with a 'b'

- The **?** wildcard replaces any one character.

*ls /usr/bin/?b**	lists all programs having a 'b' as the second letter

Ranges

- **[ ]** is used to define a range of value.

ls a[0-9] **lists all files starting with an 'a' and have a digit in second position. Also**

*ls [!Aa]**	lists all files that don't start with an 'a' or an 'A'

- {*string1,string2*}; although not just a file naming wildcard, it can be used to match the names of existing files.

ls index.{htm,html}

Quotes and escape codes

The special meaning of metacharacters can be cancelled by *escape characters*, which are also metacharacters.

The backslash (\) is called the **escape character and** cancels the meaning of all metacharacters forcing the shell to interpret them literally.

The single quotes (' ') cancel the meaning of all metacharacters except the backslash.

The double quotes (" ") are the weakest quotes but cancel most of the special meaning of the enclosed characters except the pipe (|), the backslash (\) and a variable ($var).

The back tick

Back quotes `` will execute a command enclosed. The next example defines the variable TIME using the **date** command.

```
TIME="Today's date is `date +%a:%d:%b`"
echo $TIME

Today's date is Sun:15:Jul
```

Another way of executing commands (similar to the back ticks) is to use **$()**. This will execute the enclosed command and treat it as a variable.

```
TIME=$(date)
```

Command History

To view the list of previously typed commands you can use the **bash** built-in command **history**.

```
history
1          ls
2          grep   500 /etc/passwd
```

You can recall commands by using the Up-arrow and Down-arrow on your keyboard. There are also emacs key bindings that enable you to execute and even edit these lines.

Emacs Key Bindings for Editing the Command History	
Ctrl+P	Previous line (same as Up-arrow)
Ctrl+n	Next line (same as Down-arrow)
Ctrl+b	Go back one character on the line (same as Left-Arrow)

Ctrl+f	Go forward one character on the line (Same as Right-Arrow)
Ctrl+a	Go to the beginning of the line (Same as <End>)
Ctrl+e	Go to the end of the line (Same as <Home>)

The bang **!** key can be used to rerun a command.

Example

!x	executes the latest command in the history list starting with an 'x'
!2	runs command number 2 from the **history** output
!-2	runs the command before last
!!	runs the last command
^string1^string2	run previous command and replace string1 by string2

Other commands

Aliases

You can create aliases for commands needing many arguments. The format to create an alias is

alias myprog='command [options]{arguments}'

Command completion

By pressing **TAB,** the shell will complete the commands you have started typing in.

By typing **alias** alone at the command line you will get a list of currently defined aliases.

<< is a redirection for EOF

For example

```
$cat << stop
```

will accept standard input until the keyword 'stop' is entered.

Reflection

Try out some of the Windows Commands provided in this tutorial: - http://people.uncw.edu/pattersone/121/labs/L1_MSDOS_Primer.pdf . Compare and contrast the structure of Linux commands with the Microsoft Windows commands. Do you think that it is possible to use the Linux Graphical User Interface to perform some of the functions performed on the command line? What do you think about the Linux Graphical User Interface?

Submit your comments and observations on your Wiki Page inside the http://www.colwiki.org/Reflections_and_Assignments webpage

Module summary

Summary

In this module you learned how to work with the command shell and noted the structure of any Linus command. It is important to note that the shell is not part of the kernel but works with the kernel to execute user instructions.

You also learned how to navigate your way on the command line. It is important to learn the command line as a system administrator as you will be required to perform most of your functions within the command line.

Assignment

Assignment

Type the next commands and represent the sequence of execution (if possible) using diagrams similar to the ones used in this module.

```
ls /etc ; df > /tmp/out.1
(ls /etc ; df) > /tmp/out.2

find /etc -type f  2> /dev/null | sort

tr [a-z] [A-Z] < /etc/passwd | sort > /tmp/passwd.tmp

cat /tmp/passwd.tmp | tr [A-Z] [a-z]
```

1. List all files in /usr/X11R6/bin that don't start with an x

```
ls /usr/X11R6/bin/[!x]*
```

2. The command xterm has the following options:

-bg <color>	set background
-fg <color>	set foreground
-e <command>	execute 'command' in terminal

Set a new alias such that the su command opens a new color xterm and prompts for a root password.

```
alias su=”xterm -bg orange -fg brown -e su - &”
```

Where would you store this alias on the system? ____________

3. You can encode files using uuencode. The encoded file is redirected to stdout. For example: *uuencode /bin/bash super-shell > uufile* encodes /bin/bash and will produce a file called super-shell when running uudecode against the *uufile*
.

- Mail the uuencoded /bin/bash to a local user (for this you can either use uuencode and a pipe | , or save the uuencoded output to a file *uufile* and use STDIN redirection <).
- Split the uuencoded file into 5 files:

```
uuencode /bin/bash super-shell > uufile
split –b 150000 uufile  base-name.
```

This will create files called base-name.aa, base-name.ab, etc

To get a uuencoded file with all the original data (unsplit) do

```
cat base-name.* > uufile.new
```

Finally uudecode the file and check it still works.

```
uudecode   uufile.new
```

This should create a binary file called super-shell

4. Which tool finds the full path to a binary by scanning the PATH variable? _____

Variables

1. Do the following

Assign the value 'virus' to the variable ALERT.

```
ALERT=virus
```

Verify that it is defined using the set command:

```
set |grep ALERT
```

Is ALERT listed when using env instead of set?

Next type 'bash'. Can you access the ALERT variable?

```
bash
echo $ALERT
```

NOTE the value of ALERT: ______ (is it blank?)

Type exit (or ^D) to return to your original session.

Use the export command to make ALERT a global variable.

```
export ALERT
```

Verify that it is a global (env) variable

```
env | grep ALERT
```

Notice that the variable VAR is referenced with $VAR.

(i) Rerun this command.

(ii) Rerun this command replacing CREDIT01 by $CREDIT01

3. Using appropriate quotes change your PS1 variable to include the full path to your working directory.

(Hint: the value of PS1 is [\u@ \W]\$, you only need to replace the \W by a \w)

```
PS1='[\u@\h \w ]\$ '
```

What does PS2 look like? _________

ALTERNATIVE INTERFACES

Download the Webmin software package for your distribution by visiting www.webmin.com. The download section is http://www.webmin.com/download.html . See to the left and follow the instructions for your distribution.

Access the webmin interface by opening up your browser and visiting: http://localhost:10000/. Log in with your root username and password and evaluate the functionality of this interface for your common system administrators activities.

Module 4

Linux File Management

Introduction

In this module you will learn how to:

Upon completion of this module you will be able to:

Outcomes

- Carry out Basic File Management Operations including creating. Deleting, Editing, Renaming, Copying etc file contents and folders i.e Moving around the file system;
- Find files and directories; and Work with hard and soft links.

Terminology

Hard Link:	A hard link is essentially a label or name assigned to a file. Conventionally, we think of a file as consisting of a set of information that has a single name. However, it is possible to create a number of different names that all refer to the same contents. Commands executed upon any of these different names will then operate upon the same file contents. A hard link is an exact replica on the original file
Soft Link:	A symbolic link, also termed a soft link, is a special kind of file that points to another file, much like a shortcut in Windows or a Macintosh alias. Unlike a hard link, a symbolic link does not contain the data in the target file. It simply points to another entry somewhere in the file system. This difference gives symbolic links certain qualities that hard links do not have, such as the ability to link to directories, or to files on remote computers networked through NFS. Also, when you delete a target file, symbolic links to that file become unusable, whereas hard links preserve the contents of the file.

Moving around the File System

Absolute and relative paths

A directory or a file can be accessed by giving its full pathname, starting at the root (/) or its relative path, starting from the current directory.

Absolute path: independent of the user's current directory starts with /

Relative path: depends on where the user is doesn't start with /

As in any structured file system there are a number of utilities that can help you navigate through the system. The next two commands are built-in commands.

pwd: Gives your actual position as an absolute path.

cd: The 'change directory' command

Finding Files and Directories

We will describe the **find**, **which**, **whereis** and **locate** utilities.

find

Syntax:

find <DIRECTORY> <CRITERIA> [-exec <COMMAND> {} \;]

The *DIRECTORY* argument tells **find** where to start searching and *CRITERIA* can be the name of a file or directory we are looking for.

Examples:

```
find /usr/X11R6/bin -name ¨x*¨.
find / -user 502
```

Matching lines are listed to standard out. This output can be acted upon. For example delete the file, or change the permission. The **find** tool has the build-in option **–exec** which allows you to do that. For example, remove all files belonging to user 502:

```
find / -type f -user 502 –exec rm –f {} \;
```

xargs

This tool is often thought of as a companion tool to **find**. In fact **xargs** will process each line of standard output as an *argument* for another tool. We could use **xargs** to delete all files belonging to a user with:

```
find / -type f -user 502 | xargs rm –f
```

Certain commands such as **rm** cannot deal with too long arguments. It is sometimes necessary to delete all files in a directory with

```
ls |xargs rm -f
```

Common criteria switches for **find**	
-type	specify the type of file
-name	name of the file
-user	user owner
-atime, ctime, mtime	access, creation and modified times (multiples of 24 hrs)
-amin, cmin, mmin	access, creation and modified times (multiples of 1 min)
-newer *FILE*	files newer than *FILE*

locate

Syntax:

locate <STRING>

When using **locate** all files and directories that match the *expression* are listed.

```
locate X11R
```

The search is much faster. In fact **locate** queries the **/var/lib/slocate** database. This database is kept up to date via a daily cron job which runs **updatedb.**

When running **updatedb** from the command line the **/etc/updatedb.conf** file is read to determine pruned files systems (e.g NFS) and directories (e.g /tmp)

which

Syntax:

which string

This tool will return the full path to the file called **string** by scanning the directories defined in the user's PATH variable only. As a result **which** is only used to find commands.

whereis

Syntax

whereis string

This tool will return the full path to source or binaries as well as documentation files matching **string** by scanning the PATH variable as well as a number of well known locations

Getting the most from ls

Most common options for ls	
-I	show inode
-h	print human readable sizes
-n	list UIDs and GIDs
-p	append descriptor (/=@) to list
-R	recursively display content of directories
-S	sort by file size
-t	sort by modification time (similar to -c)

-u	show last access time

Handling Directories

Making a directory with mkdir:

When making a directory you can set the permission mode with the **–m** option. Another useful option is **-p** which creates all subdirectories automatically as needed.

Example:

```
mkdir –p docs/programs/versions
```

Removing directories:

To remove a directory use either **rmdir** or **rm -r**. If you are root you may have to specify **-f** to force the deletion of all files.

Notice: rm –rf /dir1/* removes all files and subdirectories leaving dir1 empty

rm –rf /dir1/ removes all files and subdirectories including dir1

Using cp and mv

cp

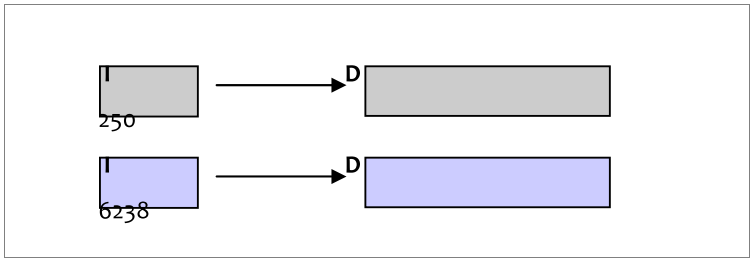
Syntax:

cp [options] *file1 file2*

cp [options] *files directory*

It is important to notice that **cp** *file1 file2* makes a new copy of *file1* and leaves *file1* unchanged.

Fig: file1 with inode 250 is copied to file2, duplicating the data to a new data area and creating a new inode 6238 for file2



You can also copy several files to a directory, using a list or wildcards. The following table lists the most used options.

Most common options for cp	
-d	do not follow symbolic link (when used with -R)
-f	Force
-I	interactive, prompt before overwrite
-p	preserve file attributes
-R	recursively copy directories

Note: cp –r /dir/* /dir2/ will copy all files and subdirectories omitting mydir

cp –r /mydir/ /dir2/ will copy all files and subdirectories including mydir

mv

Syntax:

mv [options] *oldname newname*

mv [options] source destination

mv [options] source directory

The **mv** command can both *move* and *rename* files and directories. If *oldname* is a file and *newname* is a directory then the file *oldname* is moved to that directory.

If the source and destination are on the same filesystem, then the file isn't copied but the inode information is updated to specify the new location. Most common options are **-f** forces overwrite and **-i** query interactively.

Hard and Symbolic Links

Symbolic links

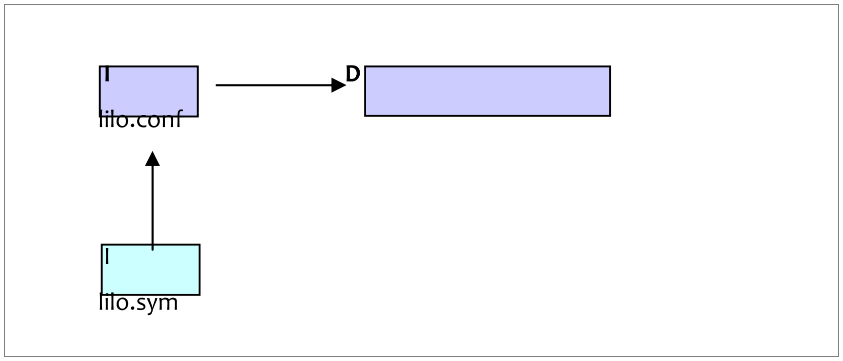
A soft link to a file or a directory creates a new inode that points to the same data area:

```
ln -s lilo.conf lilo.sym
```

This is the listing for these files. Notice that the reference count is **1** for both files.

```
-rw-------     1    root    root    223  Nov    9  09:06
lilo.conf
lrwxrwxrwx   1  root  root  9 Nov  9 09:06  lilo.sym -
> lilo.conf
```

A soft link to a file



Soft links can be created across filesystems.

Hard Links

A hard link is an additional name for the same inode and as such the reference count of the file increases by one for every new hard link.

```
ln lilo.conf lilo.link
```

In the listing notice that the reference count is **2** and that both files have the same size. In fact they are identical.

```
-rw-------  2 root  root  223  Nov  9 09:06 lilo.conf
-rw------- 2 root  root  223  Nov  9 09:06 lilo.link
```

Hard links can only be created within the same filesystem.

Touching and dd-ing

touch

Another way of creating or modifying a file is to use **touch**.

Syntax: touch {options} *file(s)*

If *file* doesn't exist it is created. You can also change the access time of a file using the **-a** option, **-m** changes the modification time and **-r** is used to apply the time attributes of another file.

Example:

touch file1.txt file2.txt	creates new files
touch myfile -r /etc/lilo.conf	myfile gets the time attributes of lilo.conf

To create a file called ***–errors*** use the **–** option:

```
touch -- -errors
```

dd

This command copies a file with a changeable I/O block size. It can also be used to perform conversions (similar to **tr**). Main options are **if=** (input file) **of=** (output file) **conv=** (conversion)

The conversion switch can be: lcase ucase ascii

Example:

```
dd if=/mnt/cdrom/images/boot.img of=/dev
```

Reflection

Evaluate the various Windows Linux File Systems including FAT and NTFS. Compare and contrast the two versions of Windows Operating System. How do they compare with the various Linux File Systems out there?

Submit your comments and observations on your Wiki Page inside the http://www.colwiki.org/Reflections_and_Assignments webpage

Module summary

Summary

In this lesson you have learned how to navigate around the Linux file system. The commands you have learned in this module are critical for your effective operations within the Linux system. You notice that you were able to change the operations of a linux command by using appropriate switches/options. This is common across all the Linux commands as this is a design principle common in all linux applications.

Assignment

Assignment

File Navigation

Make a new directory in **/tmp** called **/bin**.

```
mkdir /tmp/bin
```

In **/tmp/bin/** create a file called *newfile* (use touch, cat or vi).
Go to the root directory (cd /). View the content of *newfile* from there.
Which is the shortest command which will take you back to **/tmp/bin** ?
Which is the shortest command which will take you to your home directory ?
Is the PWD variable local or global ?

Creating and deleting directories

Which is the quickest way to make two new directories /dir1/dir2 ?
Remove the directories with **rmdir** then with **rm**.

Making space on the filesystem

In order to create more space on the device containing the directory **/usr/share/doc** we need to find a spare device with enough space and copy the contents of **/usr/share/doc** to that device. Then we create create by deleting the **/usr/share/doc** directory and creating a symbolic link point from **/usr/share/doc** to the new location.

Make a directory called **/spare** on which we will mount suitable spare devices (one of the partitions created in the previous exercises should be suitable.

```
mkdir /spare
mount <device> /spare
```

Test with **df -h /spare** and **du -hs /usr/share/doc** that the device is large enough to contain all of the existing data.

Next, copy the contents of **/usr/share/doc** to **/spare/**

```
cp -a /usr/share/doc /spare
```

Make sure the data has all been copied across then edit **/etc/fstab** to make that device available at boot time.
Delete **/usr/share/doc** and create a symbolic link pointing from **/usr/share/doc** to **/spare/doc**

```
ln -s /spare/doc /usr/share/doc
```

Do the same with **/home.** Any extra problems?

Finding Files on the System

Copy the file */etc/lilo.conf* to */etc/lilo.conf.bak*

1. Use **find** to find this new file
2. Use **locate** to find */etc/lilo.conf.bak*. (How do you update the **slocate** database ?)

Backup strategy (first step)

Find all files in your home directory that have been modified today.

find /home –mtime –1 |tee list1 |wc –-lines (-1 means less than one day)

We will introduce archiving tools later, but the output of the find command can be piped directly into **cpio**.

Module 5

Linux Post Installation Activities

Introduction

In this Module you will learn the activities that you can perform upon a basic Linux installation. More specifically, you will learn how:

- To configure a Linux Boot Manager and the Linux Boot process (From Boot Loader to Bash) ; and
- To configure peripheral devices including printers and networking

Outcomes

Terminology

LILO Boot Loader:	LILO (LInux LOader) is a generic boot loader for Linux.LILO was originally developed by Werner Almesberger. LILO does not depend on a specific file system, and can boot an operating system (e.g., Linux kernel images) from floppy disks and hard disks. One of up to sixteen different images can be selected at boot time. Various parameters, such as the root device, can be set independently for each kernel. LILO can be placed either in the master boot record (MBR) or the boot sector of a partition. In the latter case something else must be placed in the MBR to load LILO.
GRUB Boot loader:	GNU GRUB is a very powerful boot loader, which can load a wide variety of free operating systems, as well as proprietary operating systems with chain-loading. GRUB is designed to address the complexity of booting a personal computer; both the program and this manual are tightly bound to that computer platform, although porting to other platforms may be addressed in the future.
Run Levels:	The term runlevel refers to a mode of operation in one of the computer operating systems that implement Unix System V-style initialization. Conventionally, seven runlevels exist, numbered from zero to six; though up to ten, from zero to nine, may be used. S is sometimes used as a synonym for one of the levels.

We first focus on the role of the **init** program and its' associated configuration file **/etc/inittab**. The role of LILO and the GRUB Bootloader at boot time is investigated in greater depth. Finally we summarize the booting process. The document "From Power to Bash Prompt" written by Greg O'Keefe as well as the boot(7) manpage are both good references for this module.

Understanding Runlevels

Unlike most non-UNIX operating systems which only have 2 modes of functionality (on and off), UNIX operating systems, including Linux, have different runlevels such as "maintenance" runlevel or "multi-user" runlevel, etc. Runlevels are numbered from 0 to 6.

Listing - Linux runlevels

Runlevel 0 **shuts down** the machine safely, Runlevel 6 **restarts** the machine safely

Runlevel 1 is **single user** mode

Runlevel 2 is **multi-user** mode, but *does not start* NFS

Runlevel 3 is **full multi-user** mode

Runlevel 4 is not defined and generally unused

Runlevel 5 is like runlevel 3 but *runs a Display Manager as well*

Both **init** and **telinit** are used to switch from one runlevel to another. Remember that **init** is the first program launched after the kernel has been initialized at boot time. The PID for **init** is always 1.

Listing - The PID for init is always 1

```
[root@nasaspc /proc]# ps uax |grep init
USER     PID %CPU %MEM   VSZ  RSS TTY    STAT START   TIME COMMAND
root       1  0.2   0.0   1368  52  ?      S    20:17   0:04  init [3]
```

At each runlevel the system will stop or start a set of specific services. These programs are kept in **/etc/rc.d/init.d**. This directory contains *all* the services that the system may run. Once these programs are launched they will stay active until a new runlevel is called. The following services are also called daemons.

Listing - List of typicalservices (or daemons) in /etc/rc.d/init.d/

```
ls /etc/rc.d/init.d/
```

anacron	cups	identd	kadmin	krb5kdc	mcserv	nscd	random	smb	xfs
apmd	dhcpd	innd	kdcrotate	kudzu	named	ntpd	rawdevices	snmpd	xinetd
arpwatch	functions	ipchains	keytable	ldap	netfs	pcmcia	rhnsd	squid	
atd	gpm	iptables	killall	linuxconf	network	portmp	rwhod	sshd	
autofs	halt	irda	kprop	lpd	nfs	pgsql	sendmail	syslog	
crond	httpd	isdn	krb524	marsrv	nfslock	pppoe	single	tux	

Note: It is possible to stop or start manually a given daemon in /etc/rc.d/init.d by giving the appropriate argument. For example if you want to restart the **apache** server you would type:

```
/etc/rc.d/init.d/httpd restart
```

When working with runlevels you will instruct a specific predefined set of programs to run and another predefined set of programs to stop running. Say you want to be in runlevel 2, you would type

```
/sbin/init 2
```

This in turn forces **init** to read its configuration file **/etc/inittab** to find out what should happen at this runlevel.

In particular (assuming we are switching to runlevel 2) the following line in **inittab** is executed:

```
l2:wait:/etc/rc.d/rc 2
```

If you look in **/etc/inittab** the "**/etc/rc.d/rc *N***" command starts all services in the **/etc/rc.d/rc*N*.d** starting with an S and will stop of services starting with a K. These services are *symbolic links* pointing to the rc-scripts in **/etc/rc.d/init.d**.

If you don't want a process to run in a given runlevel N you can delete the corresponding symlink in /etc/rc.d/rN.d beginning with a K.

The Joys of inittab

As promised let's take a look at **/etc/inttab**. The file has the following structure:

id : runlevel : action : command

the /etc/inittab file:

```
id:3:initdefault:
# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
-----------------------snip----------------------------------
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
-----------------------snip----------------------------------
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
# Run xdm in runlevel 5
x:5:respawn:/etc/X11/prefdm –nodaemon
```

The **id** field can be anything. If a **runlevel** is specified then the **command** and the required **action** will be performed only at that specific runlevel. If no number is specified then the line is executed at *any* run level.

Recognisable features in the /etc/inittab file:

The default runlevel: this is set at the beginning of the file with the id **id** and the action **initdefault**. Notice that no command is given. This line simply tells **init** what the default runlevel is.

First program called by init: /etc/rc.d/rc.sysinit. This script sets system defaults such as the PATH variable, determines if networking is allowed, the hostname, etc ...

Default runlevel services: If the default runlevel is 3 then only the line "l3" will be executed. The action is "wait", no other program is launched until all services in run level 3 are running.

The getty terminals: The lines with id's 1-to-6 launch the virtual terminals. This is where you can alter the number of virtual terminals.

Runlevel 5: The final line in **inittab** launches the Xwindow manager if runlevel 5 is reached.

Remarks:

1. You can set a modem to listen for connections in **inittab**. If your modem is linked to /dev/ttyS1 then the following line will allow data connections (no fax) after 2 rings:

```
S1:12345:respawn:/sbin/mgetty -D -x 2 /dev/ttyS1
```

2. When making changes to **/etc/inittab** you need to force **init** to reread this configuration file. This is most easily done using:

```
/sbin/init q
```

From Boot to Bash

We can now attempt to go through the steps a Linux system goes through while booting. If an initial ram disk is specified it is loaded here. Modules are inserted from the initial ram disk. The kernel is loaded from the medium, specified in LILO's configuration. As it loads it is decompressed.

The kernel then mounts the root (/) filesystem in accordance with the configuration it receives from LILO (usually read-only). Hence essential programs in **/bin** and **/sbin** are made available. The kernel then loads **init** - the first 'userspace' process.

Init reads /etc/inittab and follows its' instructions. In particular **rc.sysinit** is run. A filesystem integrity check (f**sck**) is done on the filesystems in accordance with entries in /etc/fstab.

Next **init** goes into the default runlevel, the **gettys** start and the boot process is over.

The prompt to login is now managed by the gettys on the ttys. After the user has typed in their username and pressed return;

/bin/login is started.

The user is prompted by /bin/login for the password. The user enters a password and presses return.

The password the user is compared to the password in /etc/passwd or /etc/shadow.

Reflection

Visit http://technet.microsoft.com/en-us/library/cc739412(WS.10).aspx and consider the use of the Windows device structure. Compare this with Linux

Submit your comments and observations on your Wiki Page inside the http://www.colwiki.org/Reflections_and_Assignments webpage

Module summary

Summary

Inorder to effectively utilize Linux computing resources, most system administrators will prefer to establish "lean and mean" machines which run the necessary Linux services to be able to achieve what is required of them. This is possible through the use of runlevels. Services not required, which belong to different run levels, can be disabled or activated by changing the run levels.

In this module you have also learned how to control the boot process and customize it to your requirements.

Assignment

Assignment

Take a look at the **boot(7)** manpage, it covers most of what we did in this module.

1. Change the system's default run level to 3 and then 5.

- How do you know your current runlevel?

2. Enable the Ctrl+Alt+Del in runlevel 3 only.

3. Add a new login prompt on tty7.

- How can you force **init** to read its' configuration file?

4. Use **dmesg** to read the chipset of your ethernet card.

5. Investigate differences between **shutdown**, **halt** and **reboot**.

- Which option to **shutdown** will force an **fsck** at the next boot?

6. Use the tools **chkconfig** or **ntsysv** to disable the **sshd** daemon in runlevel 2,3,4, and 5

Verify that the symbolic links in the rc2.d, rc3.d, rc4.d and rc5.d directories have changed.

7. Reboot the system. At the boot prompt give the appropriate **init=** parameter to skip **/sbin/init** and start a simple bash session.

Module 6

Devices and Linux File System Management

Introduction

In this Module you will learn how to:

Outcomes

- Create partitions and File Systems
- File Permissions
- Monitoring Disk Usage
- Control File System Mounting and Unmounting

Terminology	**Partitions:**	A hard disk partition is a defined storage space on a hard drive. Disk partitioning is the act or practice of dividing the storage space of a hard disk drive into separate data areas known as partitions.

Creating Linux Partitions

One has the choice to associate a piece of hardware (or resource) to a directory. For example the root directory "/" which is more or less like the C:\ drive for DOS could correspond to the /dev/hda2 partition, and the subdirectory **/boot** could correspond to the partition /dev/hda3. "/dev/hda3 is said to be **mounted** on /boot". The directory on which a block device is mounted is then called a **mount point**.

While installing Linux you will have the choice of creating new partitions and associating each partition to a mount point. For advanced users this is done in two steps:

- Use the **fdisk** tool to create new partitions
- Associate a mount point to each partition

For intermediate users most distributions include a userfriendly tool that does both these steps .The very early success of RedHat over other projects such as Debian was the introduction of intuitive installation tools such as **DiskDruid.**Finally, for beginners and busy sysadmin's, the latest Linux distributions will automatically assign a partition scheme.

Once the operating system is installed you can use the **fdisk** utility to configure new partitions. We will next look at the basic syntax for **fdisk**

1) Start partitioning the first hard drive:

```
fdisk  /dev/hda
```

2) Type **m** for help. Then create a new partition with **n**.

3) To write the changes to disk type **w**.

4) REBOOT.

These four points outline the steps you would follow to create new partitions. The last point is often overlooked. This forces the partition table in the master boot record **MBR** to be reread.

NOTICE
You need to create a filesystem on a new partition with **mkfs** or **mke2fs** before using it

This ends the survey of available partitioning tools. We next take a look at bootloaders.

Managed Devices

At boot time the **/etc/fstab** file assigns mount points for block devices.

The /etc/fstab format

device	mount-point	fstype	options	dump-number	fsck-number

Sample /etc/fstab

```
LABEL=/         /              ext2    defaults             1 1
LABEL=/boot        /boot       ext2    defaults             1 2
LABEL=/home     /home          ext3    defaults             1 2
/dev/fd0                /mnt/floppy  auto      noauto,owner   0 0
LABEL=/usr              /usr           ext2    defaults
       1 2
LABEL=/var              /var           ext3    defaults
       1 2
none                    /proc          proc    defaults
       0 0
none                    /dev/shm       tmpfs   defaults
       0 0
none                    /dev/pts       devpts  gid=5,mode=620 0 0
/dev/hdc9               swap,pri=-1    swap    defaults
       0 0
/dev/cdrom         /mnt/cdrom  iso9660  noauto,owner,kudzu,ro  0
0
```

On a running system the **/etc/fstab** file also acts as a *shortcut* for assigning a resource to a specific directory. For example:

```
mount /dev/cdrom
```

The **mount** utility reads **fstab** and deduces where to mount the resource. Notice that some of the devices are accessed using a label. Labels are assigned to devices with the **tune2fs** tool:

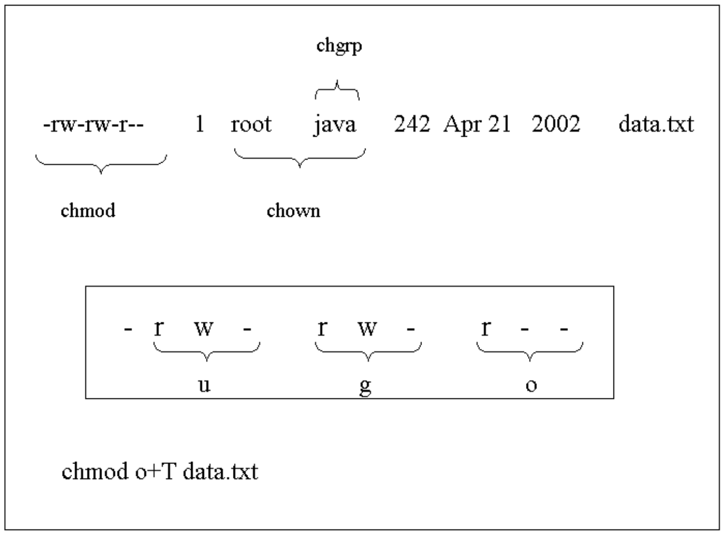
```
tune2fs -L /usr/local /dev/hdb12
```

Option summary for **mount:**	
rw,ro	read-write and read-only
users	the device can be read and unmounted by all users
user	the device can unmounted only by the user

owner	the device will change it's permission and belong to the user that mounted it
usrquota	start user quotas on the device
grpquota	start group quotas on the device

NOTICE
Remember that **mount -a** will mount all filesytems in **/etc/fstab** that have not been mounted and do not have the option **noauto**

File Permissions



Permissions can be acted upon with **chmod.** There are 3 categories of ownership for each file and directory:

The symbolic values for the **owner** fields:

u: a valid user with an entry in /etc/passwd

g: a valid group with an entry in /etc/group

o: other

Example:

```
-rw-rw-r--    1 jade   sales         24880 Oct 25 17:28 libcgic.a
```

Changing Permissions:

```
chmod g=r,o-r libcgic.a
chmod g+w  libcgic.a
```

Changing user and group:

```
chown  root  libcgic.a
chgrp  apache libcgic.a
```

NOTICE
A useful option for **chmod**, **chown** and **chgrp** is **–R** which recursively changes ownership and permissions through all files and directories indicated.

Symbolic and octal notation

Permissions can be read=r, write=w and execute=x. The octal values of these permissions are listed in the next table.

Octal and symbolic permissions.

Symolic	octal

read write	4 2
execute	1

Permissions apply to the user, the group and to others. An item has a set of 3 grouped permissions for each of these categories.

How to read a 755 or -rwxr-xr-x permission

user	group	other
rwx 4+2+1=**7**	r_x 4+1=**5**	r_x 4+1=**5**

The standard permission

UNIX systems create files and directories with standard permissions as follows:

Standard permission for:

```
Files       666         -rw-rw-rw-
Directories 777         -rwxrwxrwx
```

Umask

Every user has a defined **umask** that alters the standard permissions. The **umask** has an octal value and is subtracted(*) from the octal *standard permissions* to give the files permission (this permission doesn't have a name and could be called the file's *effective* permission).(*) While subtraction works in most cases, it should be noted that technically the standard permissions and the umask are combined as follows:

Final Permissions = Standard Permissions (logical AND) (NOT)Umask

On systems where users belong to separate groups, the umask can have a value of 002. For systems which place all users in the *users* group, the umask is likely to be 022.

SUID permissions

It is possible for root to give users permission to execute programs they would usually be unable to. This permission is the SUID permission with a symbolic value **s** or a numerical value **4000**.

For example root can write a C shell script that executes a program and set the SUID of the script with chmod 4777 script or chmod u+s script. (NB Bourne and Bash scripts do not honour SUID bits set on the script files.)

Examples:

```
chmod 4755 /bin/cat
chmod u+s /bin/grep
```

SGID permissions

The SGID is a similar permission set for group members. The symbolic value is **s** and the octal value of **2000**.

Setting SGID on a directory changes the group ownership used for files subsequently created in that directory to the directories group ownership. No need to use newgrp to change the effective group of the process prior to file creation.

Examples:

```
chmod 2755 /home/data
chmod g+s /bin/wc
```

The sticky bit

The sticky bit permission with value **1000** has the following effect:

.1. Applied to a directory it prevents users from deleting files unless they are the owner (ideal for directories shared by a group)

.2. Applied to a file this used to cause the file or executable to be loaded into memory and caused later access or execution to be faster. The symbolic value for an executable file is **t** while for a non executable file this is **T**. As file system caching is more generic and faster, file sticky bits tend not be supported any more.

chmod 1666 /data/store.txt

chmod o+t /bin/bash

Module summary

Summary

In this module you learned how to work with partitions and allocate the appropriate rights to files that you will work with. Linux is very particular on the rights issues. The appropriate ownership is required to be able to execute and run applications within Linux.

It is important that you understand how Linux maintains and manages its access rights.

Assignment

Assignment

Create 2 new partitions (larger than 50M) on the /dev/hda device using fdisk. HINT: To create a new partition type n. The partition type defaults to 83 (Linux)

- To write the new partition table type w.
- The partition table needs to be read: REBOOT the computer!

Format the first partition using the ext2 filesystem type and the second with reiserfs.

HINT: The mkfs tool is a front for mkfs.ext2 or mkfs.reiserfs, etc. The syntax is

```
mkfs –t <fstype> <device>
```

Make directories in /mnt and mount the new partitions

```
mkdir /mnt/ext2
mkdir /mnt/reiserfs
```

Use mount to verify which devices are mounted. The permissions set in fstab are visible too.

Use df to check the total number of blocks used. The –k option will convert the number of blocks in kilobytes (the default block size for ext2)

Run fsck on one of the newly created filesystems. The fsck utility is a front for fsck.ext2, fsck.ext3, fsck.reiserfs, etc. The syntax is:

```
fsck  <device>
```

Notice that there are no tools to create ext3 formated partitions. In fact the ext3 format is the same as the ext2 format with a journal added. These are the steps:

```
mke2fs /dev/hda10

tune2fs –j /dev/hda10
```

At this stage the system has added a journal to the /dev/hda10 partition, making it an ext3 formated partition. This process is non-destructive and reversible. If you mount an ext3 as an ext2 filesystem, the .journal file will be erased. You can add it again with tune2fs.

Module 7

Process Management

Introduction

In this Module you will learn about how to manage running linux processes including:

- Managing the Init Process.
- Viewing running processes
- Modifying various attributes of running processes
- Manage runlevels and system initialization from the CLI and configuration files(*for example: /etc/inittab and init command, /etc/rc.d, rc.local)*

Outcomes

Processes have a unique Process ID the **PID**. This number can be used to modify a process' priority or to stop it.

A process is any *running* executable. If process_2 has been spawned by process_1, it is called a *child* process. The spawning process_1 is called the *parent* process.

The process family tree

The **pstree** command gives a good illustration of *parent* and *child* process hierarchy.

Part of the pstree output

```
bash(1046)---xinit(1085)-+-X(1086)
              `-xfwm(1094)-+-xfce(1100)---xterm(1111)---bash(1113)-+-
pstree(1180)
                     |                    |-soffice.bin(1139)---soffice.bin(1152)-+
                                                             -soffice.bin(1153)
                     |                              |                            |-
soffice.bin(1154)
                     |                              |                            |-
soffice.bin(1155)
                     |                              |                            |-
soffice.bin(1156)
                     |                              |                            `-
soffice.bin(1157)
                     |                    `-xclock(1138)
                     |-xfgnome(1109)
                     |-xfpager(1108)
                     |-xfsound(1107)
                     `-xscreensaver(1098)
```

In the above figure all the process' PIDs are shown; these are clearly incremental. The most common used options are **-p** to display PIDs and **-h** to highlight users processes only.

Finding a running process

A more direct way to determine which processes are running is to use **ps**. Most users have a set combination of options which work for most situations. Here are three such options:

ps -ux	all processes run by the user
ps T	processes run under the current terminal by the user
ps aux	all processes on the system

It is recommended you read the **ps manpage** and choose your own best options!

***ps** accommodates UNIX-style and BSD-style arguments*

***ps** accommodates UNIX-style and BSD-style arguments*
usage: ps -[Unix98 options] ps [BSD-style options] ps --[GNU-style long options] ps –help for a command summary

Summary of options
-a show all processes for the current user linked to a tty (*except* the session leader) **-e** or **-A** show all processes **-f** gives the PPID (Parent Process ID) and the STIME (Start Time) **-l** is similar to **-f** and displays a long list **a** show all processes linked to a tty, including other users x show all processes without a controlling tty as well

Continuously updating process information

The **top** utility will update information on processes at an adjustable rate.

While **top** is running you can type **h** for a list of commands. The space bar will update information instantly.

You can also use **top** to change a process' priority as we shall see in the next section.

Modifying a running process

Stopping processes

The **kill** command can be used to send *signals* to processes. There are 63 signals available. The default signal terminates a process and is called SIGTERM with value 15.

kill

Syntax

kill SIGNAL process_PID

Every process can choose whether or not to catch a signal except for the SIGKILL which is dealt with by the kernel. Most daemons redifine the SIGHUP to mean “re-read configuration file”.

Most Common Signals

Most Common Signals

1 or SIGHUP hangup or disconnect the process

2 or SIGINT same as Ctrl+C interrupt

3 or SIGQUIT quit

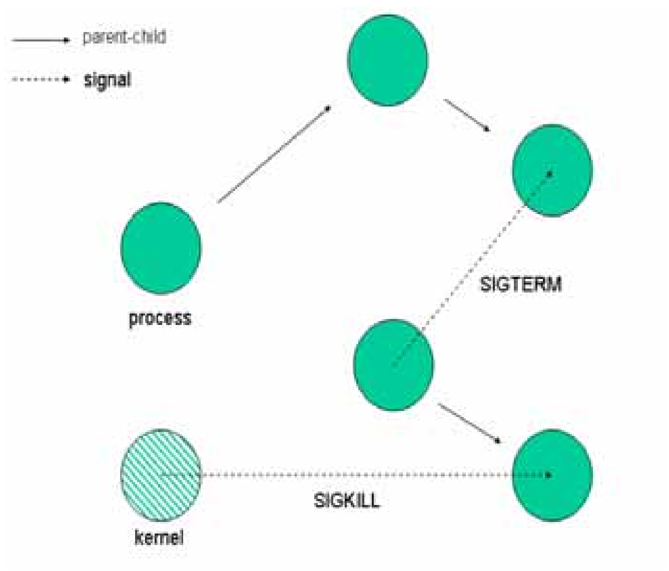
One can also stop processes without knowing the process' PID using **killall.**

killall

Syntax

killall SIGNAL process_NAME

Figure illustrating Interprocess signaling

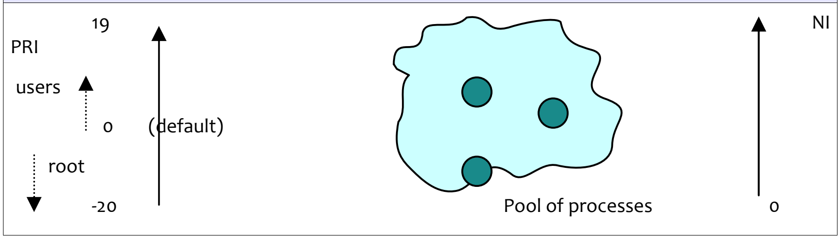


<u>Process priority and nice numbers</u>

Nice numbers (NI) alter the CPU priority and are used to balance the CPU load in a multiuser environment. Each process is started with a default nice number of **0.** Nice numbers range from **19** [lowest] to **-20** [highest].

Only root can decrease the nice number of a process. Since all processes start with a default nice number of zero as a consequence negative nice numbers can only be set by root!

nice numbers and CPU priorities



To modify a process' priority that is already running use **renice**. To set a process' priority use **nice.**

Syntax

Nice –<NI> <process>

renice <+/-NI> -p <PID>

Notice that **renice** works with PIDs and handles lists of processes at a time. A useful option to **renice** is the **-u** option which affects all processes run by a user.

Set nice number 1 for processes 234 and 765:

```
renice +1  -p 234 765
```

Set nice number -5 for **xclock**:

```
nice  --5  xclock
```

Processes and the Shell

<u>Background and forground processes</u>

After you have started a process from the shell you automatically leave the shell interpreter. You will notice that no commands will respond. The reason for this is that it is possible to run programs in the *foreground* **fg** or in the *background* **bg** of a shell.

When a program is running in the foreground it is possible to recover the shell prompt but only by interrupting the program for while. The interruption signal is **Ctrl Z.**

<u>Stopping and starting jobs</u>

A process started from a shell is also called a *job.* Once the job receives the **^Z** signal it is stopped and the shell prompt is recovered. To restart the program in the background simple type: **bg**.

Example

```
[mike localhost  /bin]$xclock  xclock running in forground, shell prompt lost
[1]+  Stopped          xclock           xclock received ^Z signal
[mike localhost  /bin]$bg  shell prompt recovered, issue the bg command
[1]+ xclock &                          xclock is running in the background
[mike localhost  /bin]$
```

Notice the [1]+ symbol above. The integer is the process' *job number,* which it can be referred to as.

The '+' sign indicates the last modified process. A '-' sign would indicate the second last modified process.

Listing jobs

The **jobs** utility lists all running processes started from the current shell. The *job number*, the job's state (running/stopped), as well as the two last modified processes, will be listed.

Output for jobs

```
[1]-  Stopped          xclock
[2]   Running          xman &
[3]+  Stopped          xload
```

The job number

One can conveniently stop and start a selection of jobs using the *job number*. This is achieved with the **fg** command.

Calling job 2 to the foreground and killing job 1

fg 2 or fg %2 or fg %?xma	kill –9 %1

Avoiding HUP with nohup

Finally there is a program called **nohup** which acts as a parent process independently from the user's session. When a user logs off, the system sends a HUP to all processes owned by that process group. For example, to avoid this HUP signal a script called **bigbang** which attempts to calculate the age of the Universe should be started like this:

```
nohup bigbang &
```

Module summary

Summary

As a system administrator, you will required to work with continuously running processes commonly reffered to as daemons. It is important to know how to troubleshoot these processes and ensure that they are available to deliver the services required to your users. What you have learned today is critical to your success as a system administrator.

Assignment

Assignment

You should run X before starting these exercises.

1. Check the current nice value of your running x-terminal. Change this value using **top** or **renice**.
2. What is the equivalent signal of a **^Z** sent to a process? (List all signals with **kill –l**)
3. Which signal is redefined for most daemons and forces the configuration file to be reread?
4. What is the default signal sent to a process, using **kill** or **killall**?
5. Which signal is directly handled by the kernel and cannot be redefined?
6. Make sure you log into a virtual terminal (tty1 to tty6) before doing this. We want to run a script that will continue to run once we logout using the **nohup** parent process.
7. In the **/tmp** directory create a file called *print-out* with the following content:

```
#!/bin/bash
 count=0
 while (true) do
     echo this is iteration number $count
     let count+=1
done
```

We first do the following (without using **nohup**) :

```
cd /tmp

./print-out &

exit
```

You may not see the command line when typing **exit** but this should log you out. When you log back in check that *print-out* is no longer running

```
ps ux |  grep  print-out
```

Next start the command with

```
nohup /tmp/print-out &
exit
```

Log back in and test these commands

```
ps ux |grep print-out
tail -f ~/nohup.out
Ctrl+C
killall print-out
ps ux|grep print-out
tail -f ~/nohup.out
```

Module 8

Groups and User Management

Introduction

Upon completion of this module you will be able to:

- Create new users
- Work with groups
- Understand the user and groups configuration files
- User management command line options.

Outcomes

Creating new users

Step 1: Create an account

The **/usr/sbin/useradd** command adds new users to the system and the symbolic link **adduser** points to it.

Syntax:

useradd [options] ***login-name***

Example: add a user with login-name rufus

```
useradd rufus
```

Default values will be used when no options are specified. You can list these values with **useradd –D**.

Default options listed with **useradd –D**

```
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
```

SKEL=/etc/skel

Notice that this information is also available in the file **/etc/default/useradd**

Step 2: Activate the account with a new password

To allow a user to access his or her account the administrator must allocate a password to the user using the **passwd** tool.

Syntax:

passwd *login-name*

These steps create a new user. This has also defined the user's environment such as a *home directory* and a *default shell*. The user has also been assigned to a group, his *primary* group.

Working with groups

Every new user is assigned to an initial (or *primary*) group. Two conventions exist.

Traditionally this *primary* group is the same for all users and is called **users** with a group id (GID) of **100**. Many Linux distributions adhere to this convention such as Suse and Debian.

The User Private Group scheme (UPG) was introduced by RedHat and changes this convention without changing the way in which UNIX groups work. With UPG each new user belongs to their own *primary* group. The group has the same name as the login-name (default), and the GID is in the 500 to 60000 range (same as UIDs).

As a consequence, when using the traditional scheme for groups the user's **umask** (see LPI 101) is set to **022**, whereas in the UPG scheme the **umask** is set to **002**.

Belonging to groups

A user can belong to any number of groups. However at any one time (when creating a file for example) only one group is the *effective* group.

The list of all groups a user belongs to is obtained with either the **groups** or **id** commands.

Example for user root:

List all ID's:

```
id
```

➔ ▶ uid=0(root) gid=0(root) groups=0(root), 1(bin), 2(daemon), 3(sys), 4(adm), 6(disk), 10(wheel), 600(sales)

List all groups:

```
groups
```

➔ ▶ root bin daemon sys adm disk wheel sales

Joining a group

Joining a group changes the user's *effective* group and starts a new session from which the user can then logout. This is done with the **newgrp** command.

Example: joining the *sales* group

```
newgrp sales
```

If the **groups** command is issued, the first group on the list would no longer be *root* but *sales*.

Creating a new group

The **groupadd** tool is used to administer groups. This will add an entry in the **/etc/group** file.

Example: Create the group devel

```
groupadd devel
```

Adding a user to a group

Administration tasks can be carried out with the **gpasswd** tool. One can add (-**a**) or remove (**-d**) users from a group and assign an administrator (**-A**). The tool was originally designed to set a single password on a group, allowing members of the same group to login with the same password. For security reasons this feature no longer works.

Example: Add rufus to the group devel

```
gpasswd -a rufus devel
```

Configurations files

The /etc/passwd and /etc/shadow files:

The names of all the users on the system are kept in **/etc/passwd**. This file has the following stucture:

1. Login name
4. Password (or x if using a shadow file)
3 The UID

The GID

5. Text description for the user
6. The user's home directory

7.The user's shell

These 7 fields are separated by colons. As in the example below.

/etc/passwd entry with encrypted passwd:

```
george:$1$K05gMbOv$b7ry0KGTd2hDrW2sT.h:Dr                G
Micheal:/home/georges:/bin/bash
```

In order to hide the encrypted passwords from ordinary users you should use a shadow file. The **/etc/shadow** file then holds the user names and encrypted passwords and is readable only by root.

If you don't have a shadow file in /etc then you should issue the following command:

```
/usr/sbin/pwconv    (passwd -> shadow)
```

This will leave an '**x**' in the 2nd field of /etc/passwd and create the /etc/shadow file. If you don't wish to use shadow passwords you can do so using

```
/usr/sbin/pwunconv   (shadow -> passwd)
```

Caution: When using a shadow password file the **/etc/passwd** file may be world readable (644) and the **/etc/shadow** file must be more restritcted (600 or even 400). Howvever when using **pwunconv** make sure to change the permissions on **/etc/password** (600 or 400).

<u>The /etc/group and gshadow files</u>:

In the same way, information about groups is kept in **/etc/group**. This file has 4 fields separated by colons.

1. Group name
2. The group password (or x if gshadow file exists)
3. The GID
4. A comma separated list of members

Example **/etc/group** entry:

```
java:x:550:jade, eric, rufus
```

As for users there is a **/etc/gshadow** file that is created when using shadow group passwords. The utilities used to switch backwards and forward from shadow to non-shadow files are as follow

/usr/sbin/grpconv	creates the /etc/gshadow file

/usr/sbin/grpunconv	deletes the gshadow file

The /etc/login.defs and /etc/skel/ files. The /etc/login.defs file contains the following information:

- the mail spool directory:

  MAIL_DIR

1. password aging controls:

   PASS_MAX_DAYS, PASS_MIN_DAYS, PASS_MAX_LEN, PASS_WARN_AGE

- max/min values for automatic UID selection in **useradd**:

  UID_MIN, UID_MAX

- max/min values for automatic GID selection in **groupadd**:

  GID_MIN, GID_MAX

- automatically create a home directory with **useradd**:

  CREATE_HOME

The /etc/skel directory contains default files that will be copied to the home directory of newly created users: **.bashrc**, **.bash_profiles**, …

Command Options

useradd (options)

-c	comment (Full Name)
-d	path to home directory
-g	initial group (GID). The GID must already exist
-G	comma separated list of supplementary groups
-u	user's UID

-s	user’s default shell
-p	password (md5 encrypted, use quotes!)
-e	account expiry date
-k	the skel directory
-n	switch off the UPG group scheme

groupadd (options)

-g	assign a GID

Modifying accounts and default settings

All available options while creating a user or a group can be modified. The **usermod** utility has the following main options:

usermod (options)

-d	the users directory
-g	the users initial GID
-l	the user's login name
-u	the user's UID
-s	the default shell.

Notice these options are the same as for **useradd**. Likewise, you can change details about a group with the **groupmod** utility. There are mainly two options:

groupmod (options)

-g	the GID
-n	the group name.

<u>Locking an account</u>

- A user's account can be locked by prefixing an exclamation mark to the user's password. This can also be done with the following command line tools:

Lock	**Unlock**
passwd –l **usermod -L**	**passwd -u** **usermod -U**

- When using shadow passwords, replace the **x** with a *
- A less useful option is to remove the password entirely with **passwd -d**.
- Finally, one can also assign **/bin/false** to the user's default shell in **/etc/passwd**.

<u>Changing the password expiry dates</u>:

By default a user's password is valid for 99999 days, that is 2739 years (default PASS_MAX_DAYS). The user is warned for 7 days that his password will expire (default PASS_WARN_AGE) with the following message as he logs in:

Warning: your password will expire in 6 days

There is another password aging policy number that is called PASS_MIN_DAYS. This is the minimum number of days before a user can change his password; it is set to zero by default.

The **chage** tool allows an administrator to change all these options.

```
Usage:  chage [ -l ] [ -m min_days ] [ -M max_days ] [ -W warn ]
 [ -I inactive ] [ -E expire ] [ -d last_day ] user
```

The first option **–l** lists the current policy values for a user. We will only discuss the **–E** option. This locks an account at a given date. The date is either in UNIX days or in YYYY/MM/DD format.

Notice that all these values are stored in the **/etc/shadow** file, and can be edited directly.

<u>Removing an account</u>:

A user's account may be removed with the **userdel** command line. To make sure that the user's home directory is also deleted use the **-r** option.

```
userdel -r jade
```

Reflection

Visit http://technet.microsoft.com/en-us/library/cc782880(WS.10).aspx and consider the implementation of Windows Security. Compare and contrast with Linux.

Submit your comments and observations on your Wiki Page inside the http://www.colwiki.org/Reflections_and_Assignments webpage

Module summary

Summary

In this module you learned have learned how to manage users and groups within a Linux system. This is a crucial part of a Linux system administrator and is essential to the security of your system to clearly define access rights. This you shall see when we look at the "Security of a Linux System"

Assignment

Assignment

1. Creating users

Use **adduser** to create a user called *tux* with user ID 600 and group ID 550

Use **usermod** to change this user's home directory.

- Does the new directory need to be created?
- Is the content of */etc/skel* copied to the new directory?
- Can the contents of the old home directory still be accessed by user *tux*?

Use **usermod** to add *tux* to the group wheel.

2. Working with groups

- Create a group called sales using **groupadd**.
- Add tux to this group using **gpasswd**.
- Login as tux and join the group sales using **newgrp**.

3. Conifiguration files

- Add a user to the system by editing /etc/passwd and /etc/group
- Create a group called share and add user tux to this group by manually

editing /etc/group

4. Modifying an Account

- Change the expiry date for user tux's account using **usermod**.
- Lock the user's account. (Use tools or edit /etc/shadow ...)
- Prevent the user from login in by changing the user's default shell to /bin/false
- Change the PASS_MAX_DAYS for user tux to 1 in /etc/shadow

5. Changing default settings

Use **useradd -D** to change the system's default settings such that every new user will be

assigned /bin/sh instead of /bin/bash. (Notice that this will change the file in /etc/defaults/)

Edit /etc/login.defs and change the default PASS_MAX_DAYS so that new users need to change their password every 5 days

Module 9

Text Manipulation

Introduction

In this module you will learn how to manipulate text in Linux. The learner will cover the following:

Outcomes

- The vi Editor: Modes, Inserting, Deleting, Copying, Searching, Saving and Undoing
- Regular Expressions
- The grep family and the sed Stream Editor
- Basic Shell Scripting and the Use of the Shell Environment

cat the Swiss Army Knife

cat the editor

The **cat** utility can be used as a rudimentary text editor as shown in the table below:

```
cat  > short-message
we are curious
to meet
penguins in Prague
     Crtl+D
```

Notice the use of Ctrl+D. This command is used for ending interactive input.

cat the reader

More commonly **cat** is used only to flush text to *stdout*. Most common options are

-n number each line of output

-b number only non-blank output lines

-A show carriage return

Example

```
cat  /etc/resolve.conf
search   mydomain.org
nameserver  127.0.0.1
```

tac reads back-to-front

This command is the same as **cat** except that the text is read from the last line to the first.

```
 tac  short-message
penguins in Prague
  to meet
  we are curious
```

Simple Tools

using head or tail

The utilities **head** and **tail** are often used to analyse logfiles. By default they output 10 lines of text. Here are the main usages.

List 20 first lines of **/var/log/messages:**

```
head -n 20 /var/log/messages
head -20  /var/log/messages
```

List 20 last lines of **/etc/aliases:**

```
tail -20  /etc/aliases
```

The **tail** utility has an added option that allows one to list the end of a text starting at a given line.

List text starting at line 25 in **/var/log/messages**:

```
tail +25 /etc/log/messages
```

Exercise: If a text has 90 lines, how would you use **tail** and **head** to list lines 50 to 65? Is there only one way to do this? Finally **tail** can continuously read a file using the **-f** option. This is most useful when you are expecting a file to be modified in real time.

Counting lines, words and bytes

The **wc** utility counts the number of *bytes*, *words*, and *lines* in files. Several options allow you to control **wc**'s output.

Options for **wc**

-l	count number of lines
-w	count number of characters or words
-c or -m	count number of bytes or characters

Remarks:

With no argument **wc** will count what is typed in *stdin*.

numbering lines

The **nl** utility has the same output as **cat -b**.

Number all lines including blanks

```
nl -ba /etc/lilo.conf
```

Number only lines with text

```
nl -bt  /etc/lilo.conf
```

replacing tabs with spaces

The **expand** command is used to replace TABs with spaces. One can also use **unexpand** for the reverse operations.

viewing binary files

There are a number of tools available for this. The most common ones are **od** (octal dump) and **hexdump**.

Manipulating Text

The following tools modify text layouts.

choosing fields and characters with cut

The **cut** utilility can extract a range of characters or fields from each line of a text.

The **–c** option is used to manipulate characters.

Syntax:

cut –c {range1,range2}

Example

```
cut –c5-10,15- /etc/password
```

The example above outputs characters 5 to 10 and 15 to end of line for each line in /etc/password.

One can specify the field delimiter (a space, a commas etc ...) of a file as well as the fields to output. These options are set with the **–d** and **–f** flags respectively.

Syntax:

```
cut -d {delimiter}  -f {fields}
```

Example

```
cut -d: -f 1,7 --output-delimiter=" " /etc/passwd
```

This outputs fields 1st and 7th of /etc/passwd delimited with a space. The default *output-delimiter* is the same as the original input delimiter. The **--output-delimiter** option allows you to change this.

joining and pasting text

The easiest utility is **paste,** which concatenates two files next to each other.

Syntax:

```
paste   text1   text2
```

With **join** you can further specify which fields you are considering.

Syntax:

```
join  -j1 {field_num}  -j2{field_num}  text1  text2            or
join  -1 {field_num}  -2{field_num}  text1  text2
```

Text is sent to stdout only if the specified fields match. Comparison is done one line at a time and as soon as no match is made the process is stopped even if more matches exist at the end of the file.

sorting output

By default, **sort** will arrange a text in alphabetical order. To perform a numerical sort use the **-n** option.

formatting output

You can modify the number of characters per line of output using **fmt**. By default **fmt** will concatenate lines and output 75 character lines.

fmt options

-w	number of characters per line
-s	split long lines but do not refill
-u	place one space between each word and two spaces at the end of a sentence

translating characters

The **tr** utility translates one set of characters into another.

Example changing uppercase letters into lowercase

```
tr '[A-B]' '[a-b]'  < file.txt
```

Replacing delimiters in **/etc/passwd:**

```
tr ':' ' ' < /etc/passwd
```

Notice: **tr** has only **two arguments!** The file is not an argument.

The Vi Editor

In most Linux distributions **vi** is the text editor of choice. It is considered an essential admin tool such as **grep** or **cat** and is found therefore in the **/bin** directory.

The vi Modes

In order to perform complex operations such as copy/paste **vi** can operate in different modes.

1. **Command mode:** This is the editing and navigation mode. Commands are often just a letter. For example use **j** to jump to the next line. As a rule of thumb if you want to perform an operation several times you can precede the command by a number. For example **10j** will jump 10 lines.
2. **Last Line (or column) Mode:** You enter this mode from the command mode by typing a colon**.** The column will appear at the bottom left corner of the screen. In this mode you can perform a simple search operation, save, quit or run a shell command.
3. **Insert Mode:** The easiest way to enter this mode while in command mode is to use **i** or **a**. This mode is the most intuitive and is mainly used to interactively enter text into a document.

The Esc key will exit the *insert mode* and return to *command mode*

Text Items

Items such as words and paragraphs are defined in *command mode* to allow editing commands to be applied to text documents without using a mouse.

Word, sentences and paragraphs

e resp. **b**	Move to the **e**nd/**b**egining of the current word
(resp.)	Move to the begining/end of the current sentence
{ resp. }	Move to the begining/end of the current paragraph
w	Similar to **e** but includes the space after the word

Beginning and End

^	Beginning of line
$	End of line
1G	Beginning of file
G	End of file

All these text items can be used to navigate through the text one word (**w**) or paragraph (**}**)at a time, go to the beginning of a line (**^**) the end of the file (**G**) etc. One can also use these text items to execute commands such as deleting and copying.

Inserting Text

When in command mode typing **i** will allow you to enter text in the document interactively. As with all other features in **vi** there are many other ways of doing this. The table below lists all possible inserting modes.

Insert Commands

a	Append text with cursor on the last letter of the line
A	Append text with cursor after last letter at the end of the line
i	Insert text at the current position
o	Insert text on a new line below
O	Insert text on a new line above
s	Delete the current letter and insert
S	Delete current line and insert

Deleting Text

If you want to delete a single character while in command mode you would use **x** and **dd** would delete the current line.

Remark: Nearly all **vi** commands can be repeated by specifying a number in front of the command. You can also apply the command to a text item (such as word., sentence, paragraph ...) by placing the entity after the command.

Deleting Text

w	single word
l	single character

<u>Examples</u>:

Delete a word:

dw

Delete text from here to the end of the current line

d$

Delete text from here to the end of the current paragraph

d}

One can simultaneously delete an item and switch to insert mode with the **c** command. As usual you can use this command together with a text item such as **w** or {.

Copy Pasting

The copy action in **vi** is the command **y** (for yank), and the paste action is **p.** If an entire line is yanked the pasted text will be inserted on the next line below the cursor.

The text selection is made with the familiar text items **w, l, }, $** etc ... There are a few exceptions such as the last example.

Examples:

Copy the text from here to the end of the current line

y$

Copy the entire current line

yy

Copy 3 lines

3yy

The latest deleted item is always buffered and can be pasted with the **p** command. This is equivalent to a cut-and-paste operation

Searching

Since searching involves pattern matching we find ourselves once again dealing with regular expressions (regex). As many UNIX text manipulation tools such as **grep** or **sed, vi** recognises regular expressions too.

To perform a search one must be in colon mode. The **/** (forward slash) command searches forward and the **?** command searches backwards.

One can also perform search and replace operations. The syntax is similar to **sed**.

Example:

Search for words beginning with 'comp' in all the text

```
/\<comp
```

Search for lines starting with the letter z

```
/^z
```

Search in the whole text for the keyword 'VAR' and replace it by 'var'

```
:% s/VAR/var
```

Undoing

At this stage is is worth mentioning that one can always undo changes (while in command mode) with the **u** command, and this as long as one hasn't saved the file yet.

Saving

The command for saving is **w**. By default the complete document is saved. One can also specify an alternative name for the file. Portions of the text can be saved to another file while other files can be read and pasted in the current document. Here are the examples which illustrate this.

Examples:

Save the current document as 'newfile'

```
:w newfile
```

Save lines 15 to 24 in a file called 'extract'

```
:w 15,24 extract
```

Read from file 'extract'. The text will be pasted at the cursor

```
:r extract
```

Warning: In the *column mode* context we have the following

.	is the current line
$	is the end of the document

Module summary

Summary

In this module you learned about how to manipulate text in the Linux command interface. As a systems administrator, you will work more with configuration files and you should be able to adequately use the exising editors to modify the operations of running processes.

You may be wondering why we studied the vi editor. This is because this is the editor that is most commonly found in any Unix like operating system including Linux.

Assessment

1. Use **cat** to enter text into a file called *message*.

```
cat >> message
line 1
^D
```

Do the same but use the keyword STOP instead of the predefined eof control (^D).

```
cat >> message << STOP
line 2
STOP
```

Next, append text to *message* using **echo**.

```
echo line 3 >> message
```

2. Create a file called *index* with two fields *REFERENCE* and *TITLE* separated by a space.

```
e.g    001    Using_Linux
```

Create a second file *pricing* with two fields *REFERENCE* and *PRICE* separated by a space

e.g 001 9.99

Use **join** to display the reference, title and prices fields.

3. Using **tr** replace all colons by semi-colons in */etc/passwd.*

Do the same using **cut.**

4. Use **head** and **tail** to list lines 70 to 85 of */var/log/messages.*

5. Use the **cut** utility together with **grep** and **ifconfig** to printout only the IP address of the first

network interface eth0.

6. In **/tmp** make a directory called *files*

```
mkdir /tmp/files
```

Create 50 files in that directory:

```
#!/bin/bash
count=0
while [ $count -lt 50 ] do
touch /tmp/files/$count.txt
let count+=1
done
```

We want to change all the **txt** extensions to **dat** extentions. For this we need to type the following on the command line:

```
for FILES in $(ls *.txt)

do

FILENAME=$(echo $FILES| cut -d. -f1)

mv  $FILES $FILENAME.dat

done
```

Module 10

The Linux Kernel

Introduction

In this module the learner will learn about:

- The Modular Linux Kernel.
- Routine Kernel Recompilation.
- Manage Kernel Modules at Runtime
- Reconfigure, Build and Install a Custom Kernel and Module

Outcomes

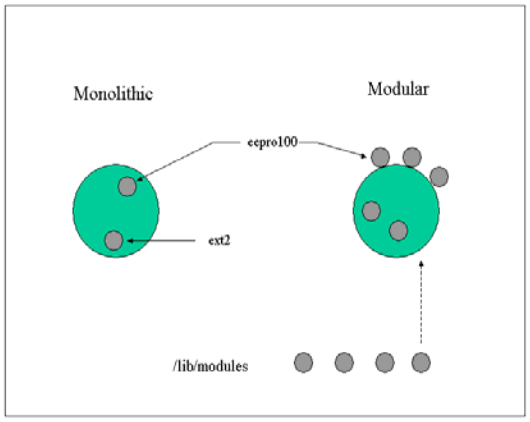
There are two types of Kernel; A monolithic and Modular Linux Kernels. These are described below:

A: **Monolithic**

A monolithic kernel is one which has support for all hardware, network, and filesystem compiled into a single image file.

B: **Modular**

A modular kernel is one which has some drivers compiled as object files, which the kernel can load and remove on demand. Loadable modules are kept in **/lib/modules**.



The advantage of a modular kernel is that it doesn't always need to be recompiled when hardware is added or replaced on the system. Monolithic kernels boot slightly faster than modular kernels, but do not outperform the modular kernel

The Modular Kernel

Many components of the Linux kernel may be compiled as modules which the kernel can dynamically load and remove as required. The modules for a particular kernel are stored in **/lib/modules/**<kernel-version>. The best components to modularise are ones not required at boot time, for example peripheral devices and supplementary file systems.

Kernel modules are controlled by utilities supplied by the ***modutils*** package:

- ***lsmod***
- ***rmmod***
- ***insmod***
- ***modprobe***
- ***modinfo***

Many modules are dependant on the presence of other modules. A flat file database of module dependencies **/lib/modules/**<kernel version>**/modules.dep** is generated by the ***depmod*** command. This command is run by the ***rc.sysinit*** script when booting the system.

-- ***modprobe*** will load any module and dependent modules listed in **modules.dep**

-- /***etc***/***modules.conf*** is consulted for module parameters (IRQ and IO ports) but most often contains a list of aliases. These aliases allow applications to refer to a device using a common name. For example the first ethernet device is always referred to as *eth0* and not by the name of the particular driver.

Sample /etc/modules.conf file:

```
alias eth0 e100
alias usb-core usb-uhc
alias sound-slot-0 i810_audio
alias char-major-108 ppp_generic
alias ppp-compress-18 ppp_mppe

# 100Mbps full duplex
options eth0 e100_speed_duplex=4
```

Routine Kernel Recompilation

Source extraction

The kernel source is stored in the ***/usr/src/linux*** directory tree, which is a symbolic link to the

/usr/src/(kernel-version) directory. When extracting a new kernel source archive it is recommended to:

- remove the symbolic link to the old kernel source directory tree

```
rm linux
```

Kernel sources which have been packaged as an RPM often create a link called **linux-2-4**

- Download and extract the source archive from http://www.kernel.org/

```
tar xjf <the name of  source archive>
```

- create a symbolic link called **linux** from the newly created directory

```
ln -s linux-2.4.20 linux
```

- The kernel is almost ready to be configured now, but first we need to make sure that all old binary files are cleared out of the source tree, and this is done with the ***make mrproper*** command.

Note: mrproper is a Scandinavian brand of cleaner that gets things "cleaner than clean", it is one step beyond "make clean".

Kernel Configuration

First edit the **Makefile** and make sure that the "EXTRAVERSION" variable is different from the existing version:

VERSION = 2

PATCHLEVEL = 4

SUBLEVEL = 20

EXTRAVERSION = *-test*

The kernel is now ready to be configured. This essentially means creating a configuration file called **.config**. This is done from the kernel source tree directory **/usr/src/linux** with any of the following

make menuconfig

make xconfig

make config

All these methods will save the configuration file as **/usr/src/linux/.config**

It is often easier to configure a new kernel using an older *.config* file by using the ***make oldconfig*** command. This will prompt the user only for new features in the kernel source tree (if the kernel is newer or has been patched).

Notice: Some distributions such as RedHat have a **configs** subdirectory containing files to be used as **.config** files with predefined configurations.

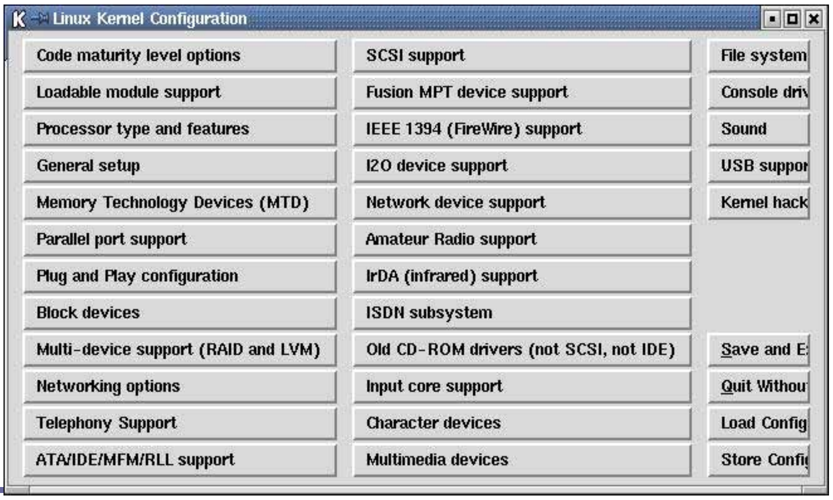
To enable kernel features (with **make menuconfig**) you will enter the top level category by moving with the arrow keys and pressing enter to access the desired category. Once in the particular category, pressing the space bar will change the kernel support for a feature or driver.

Possible support types are

- supported (statically compiled) **[*]**
- modular (dynamically compiled) **[M]**
- not supported **[ ]**

The same choices are available with the other menu editors **config** and **xconfig**.

The *make xconfig* top level menu*:*



Kernel Compilation

make dep

Once the kernel configuration is complete, it is necessary to reflect these choices in all the subdirectories of the kernel source tree. This is done with the ***make dep*** command. Files called *.depend* containing paths to header files present in the kernel source tree (/usr/src/linux/include) are generated with the **dep** target..

make clean

The **make** command gets instructions from the **Makefile** and will build what is needed. If some files are already present **make** will use them as is. In particular files with ***.o** extensions. To make sure that all the configuration options in **.config** are used to rebuild the files needed one has to run **make clean** (this deletes *.o files)

Notice: you do not need to do "make clean" at this stage if you already prepared the source directory with "make mrproper"

The kernel itself is compiled compiled with one of the commands:

make zImage

make bzImage

When the command exits without any errors, there will be a file in the **/usr/src/linux/** directory called **vmlinux**. This is the uncompressed kernel.

The two other commands will write an additional file in **/usr/src/linux/arch/i386/boot/** called **zImage** and **bzImage** respectively. These are compressed kernels using gzip and bzip2. See the next section ***Installing the New Kernel*** to find out how to proceed with these files.

make modules

The modules are compiled with ***make modules.***

make modules_install

Once the modules are compiled they need to be copied to the corresponding subdirectory in **/lib/modules**. The ***make modules_install*** command will do that. The sequence of commands are depicted in Fig below.

Kernel compilation commands:

```
make dep
make clean
make bzImage
make modules
make modules_install
```

Installing a New Kernel

The new kernel can be found in ***/usr/src/linux/arch/i386/boot/bzImage***, depending on your architecture of your system. This file must be copied to the /***boot*** directory, and named ***vmlinuz-*<full-kernel-version>**

```
/usr/src/linux/arch/i386/boot/bzImage /boot/vmlinuz-<full-kernel-version>
```

Next the **/etc/lilo.conf** or **/boot/grub/grub.conf** file needs to be edited to add our newly compiled kernel to the boot menu. Copy the "image" section from your existing kernel and add a new image section at the bottom of the file, as shown below:

Editing the /etc/lilo.conf file

```
prompt
timeout=50
message=/boot/message

image=/boot/vmlinuz
      label=linux
         root=/dev/hda6            Existing section
              read-only
------------------------------------------------------------

image=/boot/vmlinuz-<full-kernel-version>
      label=linux-new          Added section
```

```
        root=/dev/hda6
        read-only
----------snip-------------------------------
```

The symbol table for the various kernel procedures can be copied to the /boot directory:

```
cp /usr/src/linux/System.map /boot/System.map-<full-kernel-version>
```

The full kernel version

On a system, the version of the running kernel can be printed out with

uname -r

This kernel version is also displayed on the virtual terminals if the **\k** option is present in **/etc/issue**.

Initial Ramdisks

If any dynamically compiled kernel modules are required at boot time (e.g a scsi driver, or the filesystem module for the root partition) they will be loaded using an initial ramdisk.

The initial ramdisk is created with the ***mkinitrd*** command which only takes two parameters: the filename, and the kernel version number.

If you use an initial ramdisk then you will need to add an ***initrd=*** line in your */etc/lilo.conf*

Example:

```
mkinitrd /boot/initrd-$(uname -r).img $(uname -r)
```

Optional

It is recommended to copy the **/usr/src/linux/.config** file to **/boot/config-<*fiull-kernel-version*>**, just to keep track of the capabilities for the different kernels that have been compiled.

Rerunning LILO

Finally lilo needs to be run in order to update the boot loader . First ***lilo*** can be run in test mode to see if there are any errors in the configuration file:

NOTICE
The LILO bootloader needs to be updated using **lilo** everytime a changed is made in **/etc/lilo.conf**

http://widefox.pbworks.com/Kernel%20Comparison%20Linux%20vs%20Windows

Reflection

Visit http://widefox.pbworks.com/Kernel%20Comparison%20Linux%20vs%20Windows for Kernel Comparison between Windows and Linux. Note the differences. Are they significant?

Submit your comments and observations on your Wiki Page inside the http://www.colwiki.org/Reflections_and_Assignments webpage

Module summary

With the modern distributions you may not require to recompile your kernel but incase you would like to add new functionality or kernel modules, then this becomes and important topic for you.

Summary

Assignment

Assignment

Before starting with the exercises make sure you don't have an existing kernel tree in **/usr/src/**. If you do, pay attention to the /usr/src/linux symbolic link.

1. Manually recompile the kernel following the compilation steps.

- Get the **kernel**-*version***.src.rpm** package from rpmfind or a CD. Installing this package will also give you a list of dependencies, such as the **gcc** compiler or **binutils** package if they haven't yet been met.

- Install the package with **–i** (this will put all the code in /usr/src/)

- Go into the **/usr/src/linux**-*version* directory and list the **configs** directory

- Copy the kernel config file that matches your architecture into the current directory and call it .config

- Run

make oldconfig

at the command line to take into account this new .config file.

- Edit the Makefile and make sure the version is not the same as your existing kernel. You can get information on your current kernel by running **uname –a** at the command line or list the **/lib/modules** directory.

- Run

make menuconfig (or menu or xconfig)

and remove ISDN support from the kernel.

- When you exit the above program the .config file is altered but the changes have not yet taken place in the rest of the source tree. You next need to run

make dep

- Finally to force new object files (.o) to be compiled with these changes you delete all previously compiled code with

make clean

- You can now build the kernel the modules and install the modules with:

make bzImage modules modules_install

- The modules are now installed in the **/lib/modules/***version* directory. The kernel is called **bzImage** and is in the following directory:

/usr/src/linux/arch/i386/boot/

We need to manually install this kernel (2 steps):

(i)cp /usr/src/linux/arch/i386/boot/bzImage /boot/vmlinuz-<*full-kernel-version*>

(ii) That was easy! now edit **/etc/lilo.conf** and add an 'image' paragraph that will tell LILO where to find this kernel and the root filesystem.

- Run **/sbin/lilo** and reboot

2. Since we downloaded the kernel-*version*.src.rpm package we can now use this package to recompile a 'RedHat preconfigured' kernel. Notice that although no intervention is needed you won't be able to change the .config menu.

- First rebuild the compiled binary package with

rpm --rebuild kernel-*version*.src.rpm (...wait!)

- This will eventually generate the **kernel**-*version***.i368.rpm** in **/usr/src/redhat/RPMS/i386/**.

- Next, upgrade you kernel with the RPM manager using the **–U** option.

Module 11

Bash Scripting

Introduction

In this module you will learn about:

- The bash environment and bash scripting essentials
- Logical Evaluation and Loops
- Handling User Input
- Working with numbers

Outcomes

Variables

When you type a command at the prompt the bash shell will use the **PATH** variable to find which executable on the system you want to run. You can check the value of path using the echo command:

```
echo $PATH
/usr/bin:/bin:/usr/sbin:/usr/X11R6/bin:/usr/local/bin:/sbin/
:/usr/local/sbin/
```

In fact many variables are needed by the shell to accommodate for each user’s environment. For example **PWD**, **HOME**, **TERM** and **DISPLAY** are such variables.

To initialise and declare a variable the syntax is as follows:

```
VARIABLE=VALUE
```

Remember not to put any spaces around the ‘=’ sign. Once a variable is declared and initialised it can be referenced by using the dollar symbol in front as here:

```
echo $VARIABLE
```

When a shell session is started a number of configuration files are read and most of the variables are set. To free a variable from its current value use **unset**.

<u>Configuration files</u>

One can distinguish configuration files which are read at login time and configuration files which are read for each new bash session.

Login configuration files:

The files which are read at login are **/etc/profile** and **~/.bash_profile** (bash will look for alternative files too such as **~/.profile**).

Next bash will read it's runtime control files ~/**.bashrc** and (if it exists) **/etc/bashrc**.

The bashrc files:

These files are read each time a new shell session is launched (such as a new xterm). The files are **/etc/bashrc** and **~/.bashrc**.

Alias and functions can be saved in the **~/.bashrc**

<u>Function syntax</u>:

```
function-name ()
{
 command1;
 command2;
}
```

You can test which files are being read by adding an echo Profile line in **/etc/profile**, the type:

bash	No profile is read, you shouldn't see anything
bash -login	This forces bash to act as a login bash, the word Profile should show up.

The following commands control the way bash starts:

bash -norc

```
bash -noprofile
```

Notice that any new bash session will inherit the parent's global variables defined in **/etc/profile** and **~/.bash_profile**.

Scripting Essentials

A shell script is a list of instructions saved in a flat file. Only two things are necessary.

- The script's first line must be **#!/bin/bash** (for a bash script)
- The file must be readable and executable (with 755 permission for example)

If these lines are not present it is possible to run the script program by typing

bash program-name

Passing variables to the script

Variables entered at the command line are referenced inside the script as $1 for the first argument, $2 for the second, etc …

Example script, mycat:

```
#!/bin/bash
cat $1
```

This script is expecting one argument, a file, and will display the content of the file using **cat**. To run this script on the lilo.conf file, you would run:

```
./mycat /etc/lilo.conf
```

Another way of passing variables to a script is by letting the script prompt the user for input interactively. This is achieved using the **read** command. The default name of the read variable is **REPLY**. Here is the modified script:

Interactively passing:

```
#!/bin/bash
echo -n "Which file shall I display ?"
read
cat $REPLY
```

or

read -p “File to display: “ FILENAME

cat $FILENAME

Special Variables

Special variables can only be referenced and are automatically set by bash. These are the most common special variables you will encounter:

$*	List of all variables entered at the command line
$#	Number of arguments entered at the command line
$0	The name of the script
$!	PID of the most recent background command
$$	PID of the current shell
$?	Exit code of the last command

For the positional parameters $1, $2 etc … there is a **shift** operator which renames each parameter in a cyclic way as follows.

$2 becomes $1

$3 becomes $2 … etc

This can be summarised as **$(n+1) $n**

Logical evaluations

Logical statements are evaluated with the **test** command or the brackets **[ ]**. In both case the result is stored in the **$?** variable such that:

if the statement is true then	**$?** is 0
if the statement is false then	**$?** is not 0

Here are some examples to illustrate:

using **test**	using **[ ]**	Meaning
test –f /bin/bash	[-f /bin/bash]	test if /bin/bash is a file
test -x /etc/passwd	[-x /bin/passwd]	test if /etc/passwd is executable

One can evaluate more than one statement at a time using the **||** (OR) and **&&** (AND) logical operators on the command line. For example we could test if

/bin/bash is executable and in **/etc/inittab** exists:

```
test -x /bin/bash && test /etc/inittab
[ -e /bin/kbash ] || [ -f /etc/passwd ]
```

This is the same as using the flags **-o** and **-a** within the **test** operator for example

```
test -x /bin/bash -a -f /etc/inittab

[ -e /bin/kbash -o -f /etc/passwd ]
```

Logical evaluations

if then loop

```
Syntax:  if     CONDITION ; then
                  command1
                  command2
                fi
```

```
#!/bin/bash

if [ -x /bin/bash ] ; then
echo "The file /bin/bash is executable"
fi
```

if then else

```
Syntax:  if     CONDITION ; then
                  command1
                  command2
                else
                  command3
         fi
```

while loop

```
Syntax:  while CONDITION is true; do
                         command
                done
```

Example: Aligne 10 hashes (#) then exit

```
      #!/bin/bash
      COUNTER=0
      while [ $COUNTER -lt 100 ]; do
   echo  -n "#"
       sleep 1
   let COUNTER=COUNTER+1
      done
```

Until loop

```
Syntax:  until CONDITION is false; do
                     command
              done
```

Example: Same as above, notice the C style increment for COUNTER

```
      #!/bin/bash
      COUNTER=20
      until [ $COUNTER -lt 10 ]; do
   echo -n "#"
   sleep 1
   let COUNTER-=1
      done
```

for loop

```
Syntax   for VARIABLE in SET; do
                     command
              done
```

Example: For example the set 'SET' can be the lines of a file

```
#!/bin/bash
for line in `cat /etc/lilo.conf`; do
IMAGE=$(echo $line | grep image)
if [ "$IMAGE" != "" ]; then
echo Kernel configured to boot: $line
fi
done
```

Expecting user input

We assume that the script is waiting for user input, depending on the answer; the rest of the program will execute something accordingly. There are two possible ways to achieve this: **select** and **case**.

Using case

```
Syntax:  case $VARIABLE in
                 CHOICE command ;;
                 CHOICE command ;;
                 esac
```

Using select

```
Syntax:  select VARIABLE in SET; do
                         if [ $VARIABLE = CHOICE ]; then
                                 command
                         fi
                         if [ $VARIABLE = CHOICE ]; then
                                 command
                         fi
                 done
```

Working with Numbers

While shell scripts seamlessly handle character strings, a little effort is needed to perform very basic arithmetic operations.

Binary operations

Adding or multiplying numbers together can be achieved using either **expr** or the **$(( ))** construct.

expr 7 + 3; expr 2 * 10; expr 40 / 4; expr 30 – 11

Example:

$((7+3)); $((2*10)); $((40/4)); $((30-11))

Comparing values

Test operators:

Numbers	Strings
-lt	<
-gt	>
-le	<=
-ge	>=
-eq	=
-ne	!=

Reflection

Visit http://www.csie.ntu.edu.tw/~r92092/ref/win32/win32scripting.html for a "brief" tutorial on Win32 Shell Scripting. Note the significant differences between Bash and Win 32 Scripting.

Submit your comments and observations on your Wiki Page inside the http://www.colwiki.org/Reflections_and_Assignments webpage

Module summary

Summary

In many Linux distributions, tasks are automated by shell scripts. It is important for you as an administrator to know how to do shell scripting yourself.

In this module you have learned the basic ingredients to use in a shell script. The module was in no way meant to be a complete overview of all that can be done with shell script but gave you what you need to be able to start.

You can read more about bash scripting from: http://linuxreviews.org/beginner/Bash-Scripting-Introduction-HOWTO/en/index.html and http://tldp.org/LDP/abs/html/

Assignment

Assignment

1. On the command line export the variable TEST

```
export TEST=old
```

2. Write the script

```
#!/bin/bash

echo old variable: $TEST

export $TEST=new

echo exported variable: $TEST
```

3. What is the value of $TEST once the script has run?

5. The following script called test_shell will print the PID of the shell that is interpreting it

test_shell

```
#!/bin/bash

if [ -n $(echo $0 |grep test) ]; then

echo The PID of the interpreter is: $$

Else

echo The PID of the interpreter is: $$

Fi
```

5) Set the permissions to 755 and test the following commands

```
test_shell

./test_shell

bash test_shell

. test_shell

source test_shell
exec ./test_shell
```

Module 12

Software Package Installation

Introduction

Linux software is packaged in various forms. The most common forms are:

- Source files
- Packaged files including .rpm and .deb files

In this module you will learn how to:

Outcomes

Install Linux packages Linux packages from source and readymade packages including:

- Debian Packages (.deb) and using apt-get utility (command line and with synaptic)
- Red Hat Package Manager (RPM).

Terminology

Software Package:	A software package refers to computer software packaged in an archive format to be installed by a package management system or a self-sufficient installer. Linux distributions are normally segmented into packages. Each package contains a specific application or service. Examples of packages include a library for handling the PNG image format, a collection of fonts, or a web browser.The package is typically provided as compiled code, with installation and removal of packages handled by a package management system (PMS) rather than a simple file archiver.
Source file:	Source files contain source code. Source code (commonly just source) is any collection of statements or declarations written in some human-readable computer programming language. Source code allows the programmer to communicate with the computer using a reserved number of instructions. The source code which constitutes a program is usually held in one or more text files

Source file distributions

Open source projects are often distributed as tarballs (i.e compressed tarred archives). Many development environments (glade, kdevelop…) generate the files that help facilitate compiling and installation of a project.

Common Files

configure: This is a script which determines what architecture is being used. It also checks that the required compiler and libraries are present. The safest way to run the script is to use './configure'.

Makefile: This acts like a configuration file for the **make** utility. The main information provided is:

- The name of the compiler and compiling options
- The path to the shared libraries and header files
- Mapping between code files (.c) and object files (.o)

Compilation

If the files above are present then there is a good chance that you will successfully 'port' the program to your computer. Here are the routine steps:

```
./configure
make
make install
```

make install   must be run as root if the install directory is /usr/ or /usr/local

There are many options to the **./configure** script. To customise your installation you could type **./configure –help**.

An easy option is --prefix which allows you to specify the root directory for the installation. This is usually set to **/usr/local**/ by default.

Red Hat Package Manager or RPM Package Manager

RPM (RPM Package Manager) is used for managing software packages. Its main commands are rpm and rpmbuild. The powerful RPM database can be queried by the users, system administrators, and package builders for detailed information about the installed software.

Essentially, rpm has five modes: installing, uninstalling, or updating software packages; rebuilding the RPM database; querying RPM bases or individual RPM archives; integrity checking of packages; and signing packages. rpmbuild can be used to build installable packages from pristine sources. Installable RPM archives are packed in a special binary format. These archives consist of the program files to install and certain meta information used during the installation by rpm to configure the software package or stored in the RPM database for documentation purposes. RPM archives normally have the extension .rpm.

For a number of packages, the components needed for software development (libraries, headers, include files, etc.) have been put into separate packages. These development packages are only needed if you want to compile software yourself, for example, the most recent GNOME packages. They can be identified by the name extension -devel, such as the packages alsa-devel, gimp-devel, and kdelibs3-devel.

Normally, the installation of an RPM archive is quite simple: rpm package.rpm.

With this command, the package is installed, but only if its dependencies are fulfilled and there are no conflicts with other packages. With an error message, rpm requests those packages that need to be installed to meet dependency requirements. In the background, the RPM database ensures that no conflicts arise—a specific file can only belong to one package. By choosing different options, you can force rpm to ignore these defaults, but this is only for experts. Otherwise, risk compromising the integrity of the system and possibly jeopardize the ability to update the system.

The options -U or --upgrade and -F or --freshen can be used to update a package, for example, rpm -F package.rpm.

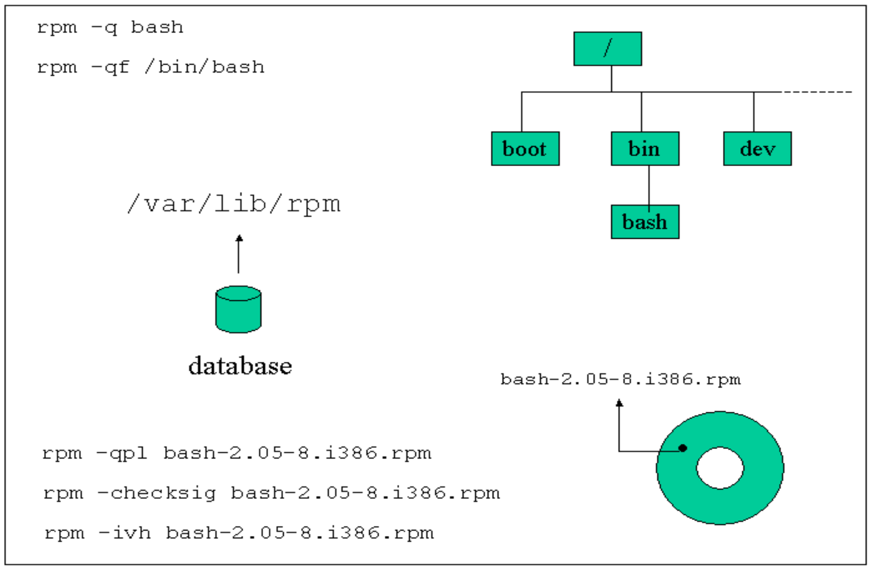

rpm -q bash
rpm -qf /bin/bash
/
boot
bin
dev
bash
/var/lib/rpm
database
bash-2.05-8.i386.rpm
rpm -qpl bash-2.05-8.i386.rpm
rpm -checksig bash-2.05-8.i386.rpm
rpm -ivh bash-2.05-8.i386.rpm

Fig - Package Manager Functions

Rpm packages generally follow the following naming convention:

name-version-release.architecture.rpm

These are the major modes for **rpm**.

Short	Long	Description
-i	–install	Installs the package
-U	–update	Updates or installs a package
-F	--freshen	Updates only installed package
-V	--verify	file size, MD5, permissions, type …
-q	--query	Queries installed/uninstalled packages, and files
-e	–erase	Uninstall package

Minior mode

Short	Description
A	applies to all installed packages
C	together with **q** lists **c**onfiguration files
D	together with **q** lists **d**ocomentation files
H	adds hashes while processing
I	together with **q** lists **i**nformation about a package
L	together with **q** lists all files and directories in a package
P	together with **q** specifies that the query is performed on the package file
V	verbose

To query a package we consider for example the package **routed-0.17.i386.rpm**. We can query this package and list its contents before installation with the **l** option as follows:

```
rpm –qpl routed-0.17.i386.rpm
```

Once this package is install we can query the installed package as with:

```
rpm –ql routed-0.17        or
rpm –ql routed
```

Finally if we want to find out which package installed the file **/usr/sbin/routed** the rpm database can be queried with:

```
rpm –qf /usr/sbin/routed
```

Three query types: uninstalled packages, installed packages and files

Query Type	Option
Package file	**-qp**
Installed package	**-q**
File	**-qf**

An extra option will allow you to get information on all installed files **–l**, documentation **–d** configuration files **–c**, etc ...

The following special options are also available

--nodeps	this allows to install without regard to dependencies
--force	force an upgrade
--test	doesn't actually install or upgrade, just prints to stdout
--requires	show package requirement

The source code for many RPM packages is also available as an RPM package and will be used to build a binary package. The naming convention is:

These packages contain at least two files, the tarball with the code and a spec

name-version-release.src.rpm

file. The spec file contains instructions to patch, compile and build the RPM package. If the code needs to be patched before compilation then the patches are included in the source package.

There are three different ways to build a RPM package. We will assume that we have a package called name-version-release.src.rpm.

Reflection

Module summary

Summary

In this module you learned you learned how to install software in Linux. You learned that software can be installed from source files and from packaged software e.g rpm or debian packages.

It is also important to also note that there exist package management software that makes it easier for you to install packaged software in linux. These include:

- **Synaptic Package Manager:** Synaptic is a computer program which is a GTK+ graphical user interface front-end to the Advanced Packaging Tool for the Debian package management system. Synaptic is usually used on systems based on deb packages but can also be used on systems based on RPM packages. It can be used to install, remove and upgrade software packages and to add repositories.

- **Yast:** Yet another Setup Tool (YaST) is an RPM-based operating system setup and configuration tool that is featured in the openSUSE Linux distribution, as well as Novell's derived commercial distributions. It features tools that can configure many aspects of the system. It is also part of the defunct United Linux. The first SuSE distribution that included YaST was released in May, 1996. YaST is free software that Novell has made available under the GPL.YaST2 is a tool for administering and maintaining a SUSE Linux installation. It allows administrators to install software, configure hardware, set up networks and servers, and more.

Assignment

Assignment

In the following examples download a source RPM file from www.rpmfind.net .

1. Installing as a tarball.

- Extract the contents of the RPM package without compiling anything with:

rpm –ivh package.rpm

- In the /usr/src/redhat/SOURCES directory, unpack the tarball with:

tar xvzf bash-2.05-8.tar.gz

- Optional (recommended!): The patches can be applied. Depending on which directory you are in the syntax will vary.

From /usr/src/redhat/SOURCES:

patch –p0 –b < *file*.patch

From /usr/src/redhat/SOURCES/bash-2.05-8

patch –p1 –b < *file*.patch

- Finally follow the usual compilation steps:

```
./configure
 make
```

If you are sure you want to install this package then make install but remember that this will not install the software using the package manager.

2. Rebuilding with the RPM package manager.

rpm –rebuild *package*.src.rpm

The compiled binary package should be in /usr/src/redhat/RPMS

- Check the package's contents with the **–qpl** option
- Install the package(s), and run queries on the installed package

- Uninstall the package

Module 13

Linux Windowing Environment

Introduction

The Linux Windowing Environment (commonly referred to as the Graphical User Interface (GUI)). The X Windows system was developed as the display component of Project Athena at the Massachusetts Institute of Technology. It is the graphical environment for UNIX. The X Window system for Linux is based on the freely distributable port of X Window version 11 release 6 (Commonly referred to as X11R6).

Outcomes

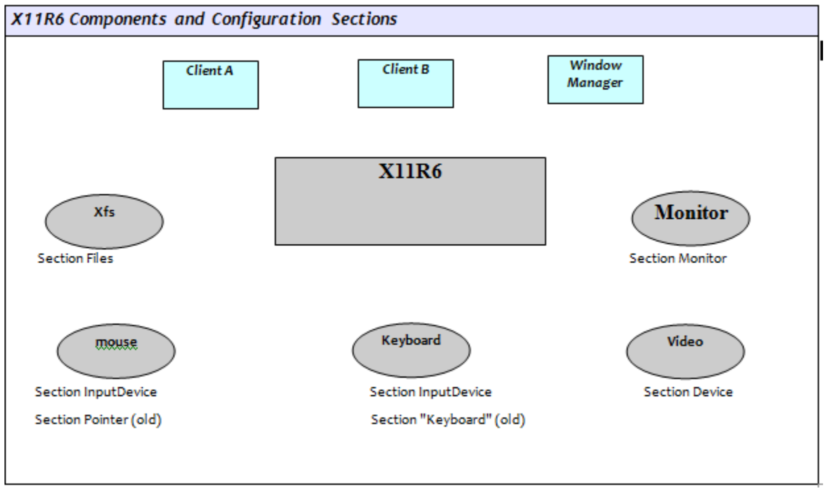
- Install and Customize a Window Manager Environment
- *Instal and Configure XFree86.*
- *Set up xdm.*
- *Identify and Terminate Runaway X Applications.*

Configuring Xfree86

This freely distributable port is commonly known as **xfree86** for the 80386/80486 and Pentium processor families. Since its initial port, Xfree86 has been ported to other computing platforms, including System V/386 and 386BSD.

The X Window System (X11) is the de facto standard for graphical user interfaces in UNIX. X is network-based, enabling applications started on one host to be displayed on another host connected over any kind of network (LAN or Internet).

Be very careful when configuring your X Window System. Never start the X Window System until the configuration is finished. A misconfigured system can cause irreparable damage to your hardware (this applies especially to fixed frequency monitors). The creators of this book and SUSE Linux Enterprise cannot be held responsible for any resulting damage. This information has been carefully researched, but this does not guarantee



The above diagram shows the components of the X11R6 server. The "Section" names refer to configuration sections in the **XF86Config** configuration file (covered in the next section). The two clients depicted on top of the server are so-called *x-applications* (e.g xclock or xterm). The window manager is also a client. Window managers add "windowing" facilities around the other x-application clients, allowing functionalities such as window dragging, focus, iconification, etc.

NOTICE:

The X11R6 server is independent from the clients that run on top. Clients are configured using specific configuration files or global files usually called **Xdefaults** or **Xresources**. The X server configuration file will only configure components such as the font server and font directories, mouse, keyboard, monitor resolution and color depth.

Configuring X11R6

Two of the configuration utilities provided with the Xfree86 software are the **XF86Setup** and **xf86config** scripts. Other vendors have specific utilities such as:

Xconfigurator, redhat-config-xfree86 (RedHat)

XFdrake (Mandrake)

sax (Suse)

The XF86Config File

All the above mentioned configuration utilities will create and edit the **XF86Config** configuration file. This file is read at start up by the X Server and determines its behaviour. This file is typically found in the /etc/X11 directory, and this is its' full path: **/etc/X11/XF86Config**. There are 11 configuration sections in the config file, they are listed below:

- ServerFlags
- Module
- InputDevice
- Device
- VideoAdapter
- Monitor
- Modes
- Screen
- ServerLayout
- DRI
- Vendor

NOTICE:

The obsolete section names *Keyboard* and *Pointer* are still recognised for compatability reasons, the new section name is now *InputDevice*

One of the first sections is the Section "Files". The FontPath keyword tells whether to get fonts from a local directory or from a font server. The RgbPath keyword is used to indicate the full path to rgb text file used to map color names to RGB notation:

```
Section "Files"
        FontPath "/path/to/fonts/dir/"
        FontPath "trans/hostname:port"
   RgbPath   "/path/to/rgb"
EndSection
```

Where trans is the transport type **unix**, hostname is the fully qualified domain name of the font server, and port is the port to connect to, usually port 7100.

Example:

```
        FontPath "unix/:7100"     # Local Font Server
        FontPath "unix/myfontserver.mydomain.com:7100"
```

Below is a sample **XF86Config** file:

```
Section "Files"
```

```
  RgbPath    "/usr/X11R6/lib/X11/rgb"
  FontPath
"/usr/X11R6/lib/X11/fonts/misc:unscaled,/usr/X11R6/lib/X11/fonts/75dpi:unscaled
,/usr/X11R6/lib/X11/fonts/100dpi:unscaled,/usr/X11R6/lib/X11/fonts/misc/"
EndSection

Section "InputDevice"
     Identifier  "Keyboard0"
     Driver     "keyboard"
EndSection

Section "InputDevice"
     Identifier  "Mouse0"
     Driver     "mouse"
     Option     "Protocol" "IMPS/2"
     Option     "Device" "/dev/psaux"
     Option     "ZAxisMapping" "4 5"
EndSection

Section "Monitor"
  Identifier     "Primary Monitor"
  VendorName     "Unknown"
  ModelName      "Unknown"
  HorizSync      31.5-37.9
  VertRefresh    55-90
  Modeline  "800x600"    40.00 800 840 968 1056 600 601 605 628 +hsync
+vsync
EndSection

Section "Device"
  Identifier     "Primary Card"
  VendorName     "Unknown"
  BoardName      "None"
  VideoRam       2048
```

```
EndSection
```

```
Section "Screen"
  Driver       "Accel"
  Device       "Primary Card"
  Monitor      "Primary Monitor"
  DefaultColorDepth 24
  BlankTime     0
  SuspendTime   0
  OffTime       0

  SubSection "Display"
    Depth      24
    Modes      "800x600"
  EndSubSection
  SubSection "Display"
    Depth      32
    Modes      "800x600"
```

Controlling X Clients

X clients are configured using the **.Xresources** or **.Xdefaults** file. These file are kept in the users home directory. It is not automatically created by default, as system-wide defaults are also available for each program.

Below is an extract from a **.Xresources**:

```
xterm_color*background: Black
xterm_color*foreground: Wheat
xterm_color*cursorColor: Orchid
xterm_color*reverseVideo: false
xterm_color*scrollBar: true
xterm_color*saveLines: 5000
xterm_color*reverseWrap: true
xterm_color*font: fixed
xterm_color.geometry: 80x25+20+20
xterm_color*fullCursor: true
xterm_color*scrollTtyOutput: off
xterm_color*scrollKey: on
```

```
term_color*VT100.Translations: #override\n\
        <KeyPress>Prior : scroll-back(1,page)\n\
        <KeyPress>Next : scroll-forw(1,page)
xterm_color*titleBar: false
```

Each of these directives is a system default directive that describes how a client will be displayed. Each line consists of the client name followed by an asterisk and the X Window parameter. Through a carefully configured .Xresources file the user can define the way a client will look each time it is started.

Starting X

An X session can be started using 2 methods:

Method 1: From the command line, after logging in onto a virtual terminal the user launches the X Server using a script called **startx**

Method 2: A Display Manager is running prompting the user with a graphical login, in **runlevel 5**.

1. From the Command Line

The **startx** script starts **xinit.** The **xinit** script has two main arguments (a) the X server and (b) the **xinitrc** script. The **xinitrc** script will source (read) the files **Xresourses** (controlling the x-applications) and the **Xclients** (choosing a window manager). So we can symbolise the startup sequence as follows:

startx --> xinit --> X -> xinitrc -> Xclient (wm/desktop)

2. Using a Display Manager

We will first describe the login. The next section covers all the functionalities of the Display Manager.

x**dm --> xlogin --> Xsession --> Xclient**

The Display Manager

There are three main display managers, xdm (generic), gdm (GNOME) and kdm (KDE). According to the LPI objectives the configuration file are in the following directories:

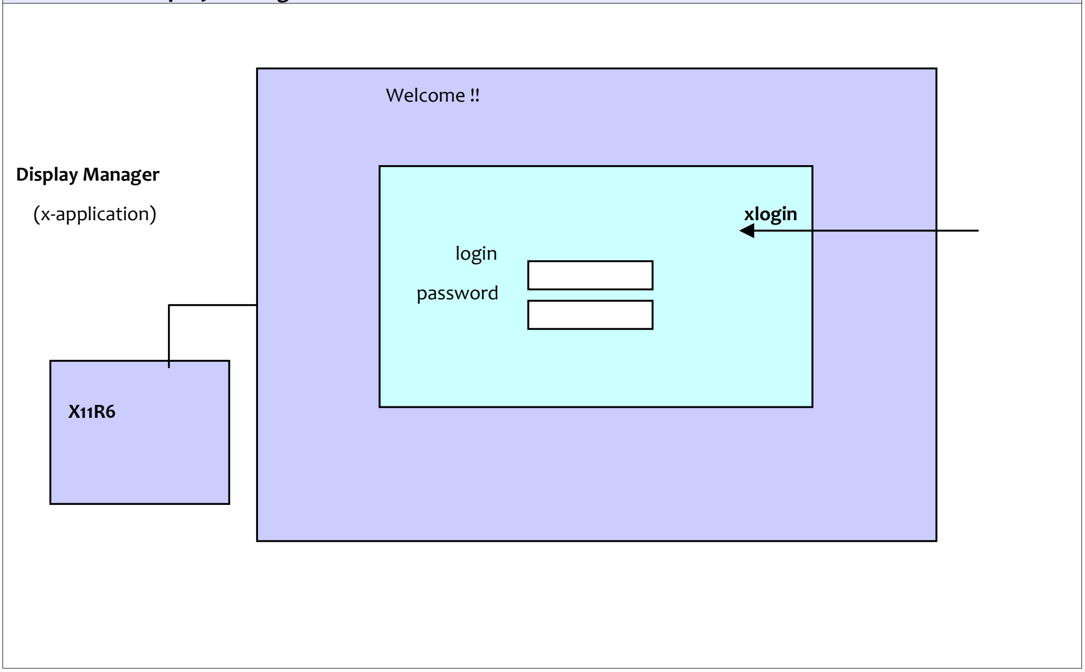
/etc/X11/xdm/

/etc/X11/gdm/

/etc/X11/kdm/

However **kdm** no longer follows this convention. So we will take a closer look at **xdm** and **gdm**. Display Managers are used mainly in run level 5 to allow local users to log onto the system using the graphical interface. However display managers can also be used to provide a graphical login interface over the network. To do this they use a protocol called **XDMCP** or X Display Manager Control Protocol. By default XDMCP is disabled (we will enable XDMCP as an exercise).

X server and Display Manager



Configuration Files

/etc/X11/xdm/Xrescources

Since the Display Manager is also an x-application, the fonts, the background colors and **xlogin**

can be configured with the **Xresourses** file in **/etc/X11/xdm/**. When using **gdm**, the

/etc/X11/gdm/Init/Default script will source **Xresources**.

/etc/X11/xdm/Xservers

This file simply maps the name of a display with an X server. For example display **:0** is understood

to be the local X server. Remember that X always runs on the first free /**dev**/**tty.**

/etc/X11/xdm/xdm-config

This is the main configuration file for **xdm.** It is also used to enable XDMCP (see Assignment)

/etc/X11/xdm/Xaccess

This file is used to enable XDMCP, allowing remote hosts to directly connect to the local server

(using **-query**) or query about other display

Reflection

Take time to navigate the Linux Graphical User Interface and compare it with the Windows or Mac OSX Operating Environment

Submit your comments and observations on your Wiki Page inside the http://www.colwiki.org/Reflections_and_Assignments webpage

Module summary

Summary

Many people see Linux as not user friendly. It is my hope that at this point in time you have noted the very good graphical capabilities of Linux. In this module you have learned how to work with the graphical user interface that is very critical to your users using Linux for the Desktop.

Assignment

Assignment

Before starting make sure you are running in runlevel **3**.

```
init 3
```

1. Log into a virtual terminal (e.g Alt+F1)

2. As root save the existing configuration file **/etc/X11/XF86Config** and try out the various configuration tools:

Redhat: Xconfigurator, redhat-config-xfree86 (8.0)

Mandrake : XFdrake

Suse: sax

XF86Setup

xf86config

X (this is the X11 server itself, use the **-configure** flag)

3. Start the X server by typing **X**. This will start X11R6 alone with no window managers. Return to a virtual terminal (e.g Ctrl+Alt+F2) and get the command line back. Then do the following:

```
export DISPLAY=localhost:0
xterm&
```

Go back into X by typing Ctrl+Alt+F7 (if you haven't changed the defaults in /etc/inittab...). You should have an xterminal running. Next type in this terminal:

```
twm&
```

What has happened? Can you kill **twm** without killing X? Go back to a

virtual terminal (e.g Ctrl+Alt+F2) and type:

```
X :1
```

Log into another virtual terminal (e.g tty3) and type:

```
export DISPLAY=:1; xterm&
```

You now have two X servers running on screen 0 and 1. How do you switch from one to another?

4. Setting up XDMCP

For this to work make sure the line containing an '*' is uncommented in **/etc/X11/xdm/Xaccess**.

If you are using **xdm** or **kdm** comment out the line in **xdm-config** as follows

```
!DisplayManager.requestPort:    0
```

This line is originally uncommented and allows only local login requests on screen **0** (more secure).

If you are using **gdm** then you will also need to edit **gdm.conf** and put

enable=true

This will turn off the default security settings for **gdm**.

If your IP is 1.2.3.4 then users on your network can start an X session with:

```
X –query 1.2.3.4 :1
```

or

```
X -indirect 1.2.3.4
```

Module 14

Linux System Administration

Introduction

In this module you will learn about:

- Logfiles and configuration files
- Log Utilities
- Automating Tasks
- Backups and Compressions
- Linux Help and Documentation

Outcomes

We will concentrate on the main tasks of system administration such as monitoring log files, scheduling jobs using **at** and **cron**. This also includes an overview of the documentation available (**manpages** and online resources) as well as some backup concepts.

Logfiles and Configuration files

The /var/log/ directory

This is the directory where most logfiles are kept. Some applications generate their own log files (such as squid or samba). Most of the system logs are managed by the **syslogd** daemon. Common system files are :

cron	keeps track of messages generated when **cron** executes
mail	messages relating to **mail**
messages	logs all messages except private authentication authpriv, cron, mail and news
secure	logs all failed authentications, users added/deleted etc

The most important log file is **messages** where most activities are logged.

The /etc/syslog.conf file

When **syslogd** is started it reads the **/etc/syslog.conf** configuration file by default. One can also start **syslogd** with **-f** and the path to an alternative config file. This file must contain a list of items followed by a priority, followed by the path to the log-file:

```
item1.priority1 ; item2.priority2        /path-to-log-file
```

Valid items are :

auth and **authpriv**	user general and private authentication
cron	cron daemon messages
kern	kernel messages
mail	
news	
user	user processes
uucp	

Valid priorities are: (from highest to lowest)

emerg

alert

crit

err

warning

notice

info

debug

*

none

Priorities are *minimal*! All higher priorities will be logged too. To force a priority to be **info** only you need to use an **'='** sign as in:

```
user.=info        /var/log/user_activity
```

Listing of /etc/syslog.conf

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                  /dev/console
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;news.none;authpriv.none                  /var/log/messages
 # The authpriv file has restricted access.
authpriv.*                               /var/log/secure
 # Log all the mail messages in one place.
mail.*                                   /var/log/maillog
 # Log cron stuff
cron.*                                   /var/log/cron
 # Everybody gets emergency messages, plus log them on another
# machine.
*.emerg                                   *
*.emerg                                  @10.1.1.254
  # Save boot messages also to boot.log
local7.*                                 /var/log/boot.log
 #
news.=crit                               /var/log/news/news.crit
news.=err                                /var/log/news/news.err
news.notice                              /var/log/news/news.notice
```

Log Utilities

The logger command

The first utility **logger** conveniently logs messages to the /var/log/messages file:

If you type the following:

```
logger  program myscipt ERR
```

The end of **/var/log/messages** should now have a message similar to this:

```
Jul 17 19:31:00 localhost penguin: program myscript ERR
```

local settings

The **logger** utility logs messages to /var/log/messages by default. There are local items defined that can help you create your own logfiles as follows. **local0** to **local7** are available items for administrators to use. The availability depends on the system (RedHat **local7** logs boot-time information in /var/log/boot.log). Add the following line to **/etc/syslog.conf**:

```
local4.*            /dev/tty9
```

Restart the **syslogd**

```
killall -HUP syslogd
```

The next command will be logged on the /dev/tty9

```
logger -p local4.notice  "This script is writing to /dev/tty9"
```

An interesting device is the /dev/speech this is installed with the Festival tools.

logrotate

The log files are updated using **logrotate**. Usually **logrotate** is run daily as a cron job. The configuration file **/etc/logrotate.conf** contains commands to create or compress files.

Listing of logrotate.conf

```
# rotate log files weekly

weekly

# keep 4 weeks worth of backlogs

rotate 4
```

```
# send errors to root

errors root

# create new (empty) log files after rotating old ones

create

# uncomment this if you want your log files compressed

compress

# RPM packages drop log rotation information into this directory

include /etc/logrotate.d

# no packages own lastlog or wtmp -- we'll rotate them here

/var/log/wtmp {

   monthly

   create 0664 root utmp

   rotate 1

}
```

Automatic Tasks

Using cron

The program responsible for running crons is called **crond.** Every minute the **crond** will read specific files containing command to be executed. These files are called *crontabs*.

User crontabs are in **/var/spool/cron/**<username>. These files should not be edited directly by non-root users and need to be edited using the **crontab** tool (see below).

The system crontab is **/etc/crontab.** This file will periodically exectute all the scripts in **/etc/cron.*** this includes any symbolic link pointing to scripts or binaries on the system.

To manipulate **cron** entries one uses the **crontab** utility. Scheduled tasks are view with the **-l** option as seen below:

```
crontab -l

➔    # DO NOT EDIT THIS FILE - edit the master and reinstall

        # (/tmp/crontab.1391 installed on Tue Jul 17 17:56:48 2001)

   # (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)

        0 * * 07 2 /usr/bin/find /home/penguin -name core -exec rm {} \;
```

Does the user **root** have any crontabs? Similarly the **-e** option will open your default editor and lets you enter a cron entry. User **root** can use the **-u** to view and change any user's cron entries. To delete your crontab file use **crontab -r**.

This is the format for crontabs :

Minutes(0-59) Hours(0-23) Day of Month(1-31) Month(1-12) Day of Week(0-6) command

Permissions:

By default any user can use **crontab**. However you can control the accessibility with **/etc/cron.deny** and **/etc/cron.allow**.

Scheduling with "at"

The **at** jobs are run by the **atd** daemon. At jobs are spooled in **/var/spool/at/.** The **at** command is used to schedule a one off task with the syntax

at [time]

Where time can be expressed as:

now

3am + 2days

midnight

10:15 Apr 12

teatime

For a complete list of valid time formats see /usr/share/doc/at-xxx/timespec. You can list commands that are scheduled with **atq** or **at -l**. The **at** jobs are saved in /var/spool/at/:

```
ls /var/spool/at/

➔     a0000100fd244d  spool
```

When using **atq** you should have a list of jobs proceeded by a number. You can use this number to dequeue it:

```
atq

➔    1     2001-07-17 18:21 a root
```

From the **atq** listing we see that the job number is **1**, so we can remove the job from the spool as follows:

```
at -d 1
```

Permissions:

By default **at** is restricted to the root user. To override this you must either have an empty **/etc/at.deny**

or have a **/etc/at.allow** with the appropriate names.

Backups and Compressions

Backup strategies

There are three main strategies to back up a system:

1. *Full*: copy all files
2. *Incremental*: The first incremental copies all files added or changed since the last full backup, and subsequently copies all the files added or changed since the last incremental backup
3. *Differential*: Copies all files added or changed since the last full backup

Example: If you made a full backup and 3 differential backups before a crash, how many tapes would you need to restore ?

Creating archives with tar

The main option to create an archive with **tar** is **-c**. You can also specify the name of the archive as the first argument if you use the **-f** flag.

```
tar -cf home.tar /home/
```

If you don't specify the file as an argument **tar -c** will simply output the archive as standard output:

```
tar -c /home/ > home.tar
```

Extracting archives with tar

Extracting is straight forward. Replace the **-c** flag with an **-x**. This will cause the archive file to create directories if necessary and copy the archived files in your current directory. To redirect the output of the extracted archive into the directory /usr/share/doc, for example, you can do:

```
tar xf backeddocs.tar -C /usr/share/doc
```

Compressions

All archives can be compressed using different compression utilities. These flags are available when creating, testing or extracting an archive:

tar option	**compression type**
Z	compress
z	gzip
j	bzip2.

The cpio utility

The **cpio** utility is used to copy files to and from archives. List of files must be given to **cpio** either through a pipe (as when used with find) or via a file redirection such as with;

- Extract an archive on a tape:

```
cpio -i < /dev/tape
```

- Create an archive for the /etc directory:

```
find /etc | cpio -o > etc.cpio
```

Documentation

Manpages and the whatis database

The manpages are organised in sections	
NAME	the name of the item followed by a short one line description.
SYNOPSYS	the syntax for the command
DESCRIPTION	a longer description
OPTIONS	a review of all possible options and their function
FILES	files that are related to the current item (configuration files etc)
SEE ALSO	other manpages related to the current topic

These are the main sections one would expect within a manpage.

The **whatis** database stores the NAME section of all the manpages on the system. This is done through a daily **cron**. The **whatis** database has the following two entries:

name(key) – one line description

The syntax for **whatis** is:

whatis <string>

The output is the full NAME section of the manpages where *string* matched *named(key)* . One can also use the **man** command to query the whatis database. The syntax is

man -k <string>

Unlike whatis this will query both the “name” and the “one line description” entries of the database. If the string matches a word in any of these fields the above query will return the full NAME section.

Example: (the matching string has been highlighted)

```
whatis lilo

lilo           (8)  - install boot loader

lilo.conf [lilo]    (5)  - configuration file for lilo
```

```
man -k  lilo

grubby        (8)  - command line tool for configuring grub, lilo, and elilo

lilo         (8)  - install boot loader

lilo.conf [lilo]  (5)  - configuration file for lilo
```

The FHS recommends manpages to be kept in **/usr/share/man**

Manpage Sections	
Section 1	Information on executables
Section 2	System calls, e.g mkdir(2)
Section 3	Library calls, e.g stdio(3)

Manpage Sections	
Section 4	Devices (files in /dev)
Section 5	Configuration files and formats
Section 6	Games
Section 7	Macro packages
Section 8	Administration commands
Section 9	Kernel routines

To access a specific section *N* one has to enter:

man *N* command

Examples:

```
man mkdir

man 2 mkdir
```

```
man crontab

man 5 crontab
```

Info pages

The FHS recommends info pages be kept in **/usr/share/info**. These pages are compressed files that can be read with the **info** tool.

The original GNU tools used info pages rather than manpages. Since then most info pages have been rewritten as manpages. However information about GNU projects such as **gcc** or **glibc** is still more extensive in the info pages compared to the manpages.

Online documents

GNU projects include documents such as a FAQ, README, CHANGELOG and sometimes user/admin guides. The formats can either be ASCII text, HTML, LateX or postscript.

These documents are kept in the **/usr/share/doc/** directory.

HOWTOs and The Linux Documentation Project

The Linux Documentation Project provides many detailed documents on specific topics. These are structured guides explaining concepts and implementations. The website URL is www.tldp.org.

The LDP documents are freely redistributable and can be contributed too using a GPL type licence.

Usenet News Groups

The main newsgroups for Linux are the **comp.os.linux.*** groups (e.g comp.os.linux.networking, comp.os.linux.security ...). Once you have setup a news reader to connect to a news server (usually available through an ISP or a University campus) one downloads a list of all existing discussion groups and subscribes/unsubscribes to a given group.

There are many experienced as well as new users which rely on the newsgroups to get information on specific tasks or projects. Take the time to answer some of these questions if you feel you have the relevant experience.

NOTICE

The **man -k** option queries both fields in the **whatis** database. This will find everything about a given item. There is a tool called **apropos** (meaning *about*) which will do the same thing as **man -k**.

Module summary

Summary

In this module you learned how to monitor linux logs and how you can perform system administration functions like back-ups and have access to Linux documentation. There is much more Linux documentation on the internet and I would recommend that if you get stack in anything you are doing with Linux you should first consult the internet by searching through discussion forums and online how-tos.

Assignment

Assignment

Logging

1. Change /etc/syslog.conf to output some of the logs to /dev/tty9 (make sure you restart **syslogd** and that the output is properly redirected)

2. Add a custom local5 item with critical priority to /ect/syslog.conf and direct the output to /dev/tty10. Restart syslogd and use logger to write information via local5.

3. Read the **/etc/rc.d/init.d/syslog** script and change **/etc/sysconfig/syslog** to allow remote hosts to send log outputs.

Scheduling

4.Create a cron entry which starts xclock every 2 minutes. Remember that **cron** is unaware of system variables such as **PATH** and **DISPLAY**.

5. Use **at.**to start xclock in the next five minutes.

Archiving

6. Use **find** to list all files that have been modified during the past 24 hours.

(hint: Redirect the output of find -mtime –1 to a file)

7.Use **cpio** to create an archive called Incremental.cpio.

(ans: Use the file created above and do **cat** FILE | **cpio –ov** > Incremental.cpio)

8 Use **xargs** and **tar** to create an archive of all files last accessed or changed 5 mins ago.

9. Do the same using the **–exec** option to **find**. Note that the files listed by **find** can be referenced by the {} symbol.

10. Extract the archive you have just created.

Module 15

Linux Networking Configuration

Introduction

This module will cover the implementation of Network configuration in linux including Network Interface Notation, Host configuration, Start and Stop Networking, Routing and Troubleshooting Network connections in Linux

Upon completion of this module you will be able to:

Outcomes

- Configure Linux Network access.
- Start and Stop Networking Services
- Configure Linux Routing
- Troubleshooting Network Connections

Terminology

Port:	In the TCP and UDP protocols used in computer networking, a port is a special number present in the header of a data packet that is used to map data to a particular process running on a computer.
Socket:	In computer networking, an Internet socket (or commonly, a network socket or socket) is the endpoint of a bidirectional communication flow across an Internet Protocol-based computer network, such as the Internet. Internet sockets (in plural) are an application programming interface (API) in an operating system, used for inter-process communication. Internet sockets constitute a mechanism for delivering incoming data packets to the appropriate application process or thread, based on a combination of local and remote IP addresses and port numbers. Each socket is mapped by the operational system to a communicating application process or thread.

Configuring Linux Networking

The network interface card (NIC) must be supported by the kernel. To determine which card you are using you can get information from **dmesg**, **/proc/interrupts**, **/sbin/lsmod**. or **/etc/modules.conf**

Example:

```
dmesg
```

```
►   Linux Tulip driver version 0.9.14 (June 25, 2009)
    PCI: Enabling device 00:0f.0 (0004 -> 0007)
    PCI: Found IRQ 10 for device 00:0f.0
    eth0: Lite-On 82c168 PNIC rev 32 at 0xf800, 00:A0:CC:D3:6E:0F, IRQ 10.
    eth0:  MII transceiver #1 config 3000 status 7829 advertising 01e1.
```

```
cat /proc/interrupts
    0:   8729602    XT-PIC  timer
    1:         4    XT-PIC  keyboard
    2:         0    XT-PIC  cascade
    7:         0    XT-PIC  parporto
    8:         1    XT-PIC  rtc
   10:    622417    XT-PIC  etho
   11:         0    XT-PIC  usb-uhci
```

```
14:    143040      XT-PIC  ide0
15:      180       XT-PIC  ide1
```

```
/sbin/lsmod
   Module            Size  Used by
   tulip           37360   1 (autoclean)
```

From the example above we see that the Ethernet card's chipset is Tulip, the i/o address is 0xf800 and the IRQ is 10. This information can be used either if the wrong module is being used or if the resources (i/o or IRQ) are not available.

This information can either be used to insert a module with a different i/o address (using the modprobe or insmod utilities) or can be saved in /etc/modules.conf (this will save the settings for the next bootup).

Host Information

The following files are used to store networking information.

- **/etc/resolv.conf** contains a list of DNS servers

```
nameserver 192.168.1.108
nameserver 192.168.1.1
search linuxit.org
```

- **/etc/HOSTNAME** is used to give a name to the computer
- One can also associate a name to a network interface. This is done in differently across distributions.
- **/etc/hosts** contains your machine's IP number as well as a list of known hosts

```
# Do not remove the following line, or various programs

# that require network functionality will fail.

127.0.0.1      localhost  localhost.localdomain

# other hosts

192.168.1.108  mesa  mesa.domain.org

192.168.1.119  pico
```

- **/etc/sysconfig/network** defines if networking must be started. (can also contain the HOSTNAME variable)

```
NETWORKING=yes

HOSTNAME=mesa.domain.org

GATEWAY=192.168.1.1

GATEWAYDEV=
```

- **/etc/sysconfig/network-scripts/ifcfg-eth0** The configuration parameters for eth0

```
DEVICE=eth0

BOOTPROTO=none

BROADCAST=192.168.1.255

IPADDR=192.168.1.108

NETWORK=192.168.1.0

ONBOOT=yes
```

```
USERCTL=no
```

Stop and Start Networking

From the command line

The main tool used to bring up the network interface is **/sbin/ifconfig**. Once initialised the kernel module aliased to eth0 in **/etc/modules.conf** (e.g tulip.o) is loaded and assigned an IP and netmask value.

As a result the interface can be switched on and off without loosing this information as long as the kernel module is inserted.

Examples: Using **ifconfig**.

```
/sbin/ifconfig eth0 down

/sbin/ifconfig eth0 up
```

```
/sbin/ifconfig eth0 192.168.10.1 netmask 255.255.128.0
```

Another tool is **/sbin/ifup**. This utility reads the system's configuration files in **/etc/sysconfig/** and assigns the stored values for a given interface. The script for **eth0** is called **ifcfg-eth0** and has to be configured. If a boot protocol such as DHCP is defined then **ifup** will start the interface with that protocol.

Examples: Using **ifup**.

```
/sbin/ifup eth0

/sbin/ifup ppp0

/sbin/ifdown eth0
```

●. Using the network script

At boot time the ethernet card is initialised with the **/etc/rc.d/init.d/network** script. All the relevant networking files are sourced in the **/etc/sysconfig/** directory.

In addition the script also reads the **sysctl** options in **/etc/sysctl.conf**, this is where you can configure the system as a router (allow IP forwarding in the kernel). For example the line:

```
net.ipv4.ip_forward = 1
```

will enable ip forwarding and the file **/proc/sys/net/ipv4/ip_forward** will contain a one.

The **network** script is started with the following command

```
/etc/rc.d/init.d/network restart
```

●. Renewing a DHCP lease

The following tools can query the DHCP server for a new IP:

pump

dhcpclient

A client daemon exists called **dhcpcd** (do not confuse this with the DHCP server daemon **dhcpd**)

Routing

Add a static route to the network 10.0.0.0 through the device eth1 and use 192.168.1.108 as the gateway for that network:

```
/sbin/route add -net 10.0.0.0 gw 192.168.1.108 dev eth1
```

A noticeable difference when using **ifup** is the system's routing table. This is because either the **/etc/sysconfig/network** file is read, where a **default gateway** is stored, or the DHCP server has sent this information together with the IP number. The routing tables are configured, checked and changed with the **/sbin/route** tool.

Routing examples:

Add a default gateway:

```
/sbin/route add default gw 192.168.1.1 eth0
```

Listing the kernel routing table:

```
/sbin/route -n

¤ Kernel IP routing table

  Destination   Gateway         Genmask        Iface

  192.168.1.0    0.0.0.0         255.255.255.0  eth0

  10.1.8.0       192.168.1.108   255.0.0.0      eth1

  127.0.0.0      0.0.0.0         255.0.0.0      lo

  0.0.0.0        192.168.1.1     0.0.0.0        eth0
```

Default Gateway**:**

In the last listing, the Destination field is a list of networks. In particular, 0.0.0.0 means ‘anywhere’. With this in mind, there are two IP’s in the Gateway field. Which one is the default gateway?

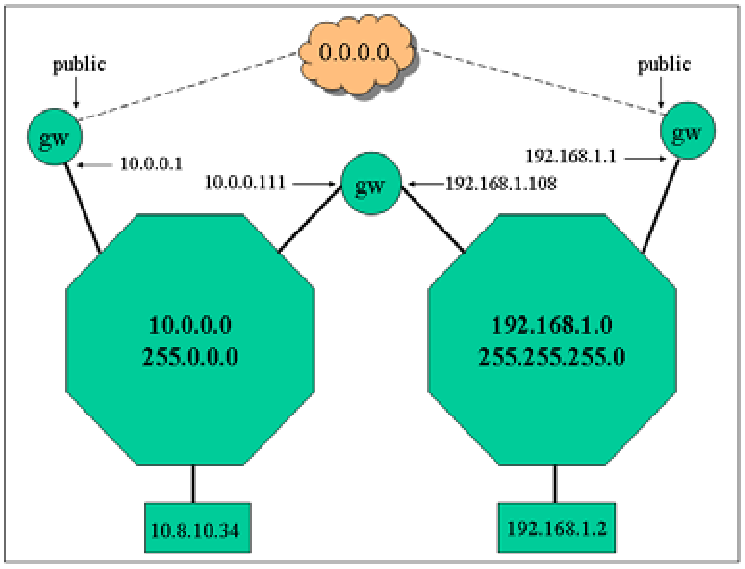
To avoid having to enter static routes by hand special daemons gated or routed are run to dynamically update routing tables across a network

If you belong to the 192.168.10.0 network and you add a route to the 192.168.1.0 network you may find that machines in the latter network are not responding. This is because no route has been set from the 192.168.1.0 network back to your host!! This problem is solved using dynamic routing.

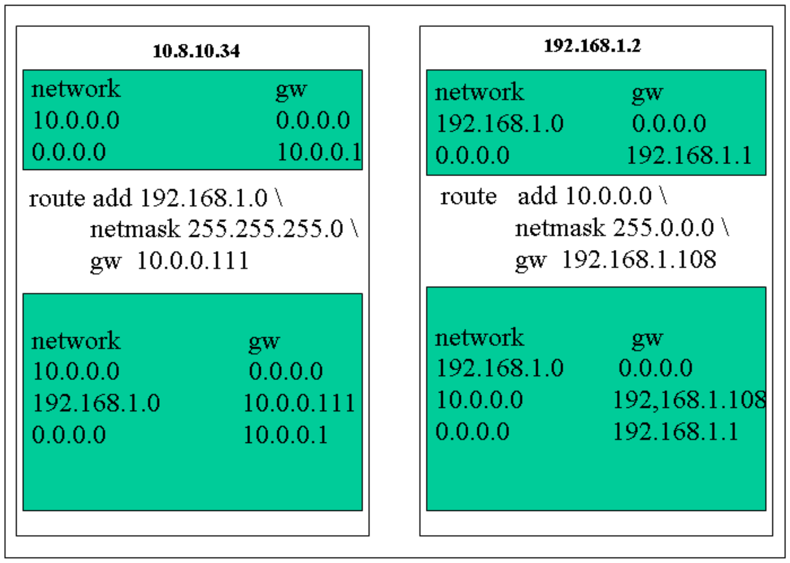
Permanent Static Routes

If you have several networks with more than one gateway you can use the **/etc/sysconfig/static-routes** (instead of routing daemons). These routes will be added at boot time by the **network** script.

A routing scenario:



Routing possibility



10.8.10.34

network	gw
10.0.0.0	0.0.0.0
0.0.0.0	10.0.0.1

```
route add 192.168.1.0 \
        netmask 255.255.255.0 \
        gw  10.0.0.111
```

network	gw
10.0.0.0	0.0.0.0
192.168.1.0	10.0.0.111
0.0.0.0	10.0.0.1

192.168.1.2

network	gw
192.168.1.0	0.0.0.0
0.0.0.0	192.168.1.1

```
route   add 10.0.0.0 \
          netmask 255.0.0.0 \
          gw  192.168.1.108
```

network	gw
192.168.1.0	0.0.0.0
10.0.0.0	192,168.1.108
0.0.0.0	192.168.1.1

Common Network Tools

Here is a short list of tools helpful when trouble shouting network connections.

ping host**:**

This tool sends an ICMP ECHO_REQUEST datagram to a host and expects an ICMP ECHO_RESPONSE.

Options for **ping**:

-b	ping a broadcast address
-c N	send N packets
-q	quiet mode: display only start and end messages

netstat:

You may get information on current network connections, the routing table or interface statistics depending on the options used.

Options for **netstat**:

-r	same as /sbin/route
-I	display list of interfaces
-n	don't resolve IP addresses
-p	returns the PID and names of programs (only for root)
-v	verbose
-c	continuous update

<u>Example</u>: Output of netstat –-inet –n :

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address      Foreign Address       State
tcp       0      0 192.168.1.10:139   192.168.1.153:1992    ESTABLISHED
tcp      0     0 192.168.1.10:22    192.168.1.138:1114     ESTABLISHED
tcp      0     0 192.168.1.10:80    192.168.1.71:18858     TIME_WAIT
```

In the above listing you can see that the local host has established connections on ports 139, 22 and 80.

arp:

Display the kernel address resolution cache.

```
arp

¤     Address       HWtype  HWaddress         Iface

        192.168.1.71    ether   00:04:C1:D7:CA:2D    eth0
```

Example:

traceroute:

Displays the route taken from the local host to the destination host. Traceroute forces intermediate routers to send back error messages (ICMP TIME_EXCEEDED) by deliberately setting the tty (time to live) value too low. After each TIME_EXEEDED notification **traceroute** increments the tty value, forcing the next packet to travel further, until it reaches its' destination.

Example:

```
CMD:                    /usr/sbin/traceroute -n  www.redhat.com

¤         traceroute: Warning: www.redhat.com has multiple addresses; using
          216.148.218.197

          traceroute to www.redhat.com (216.148.218.197), 30 hops max, 38
          byte packets

          -w sec     set the timeout on returned packets to sec

1  192.168.1.1  0.440 ms  0.347 ms  0.341 ms

          ---- snip ---

14  12.122.2.145  112.116 ms  110.908 ms  112.002 ms

15  12.122.2.74  156.629 ms  157.028 ms  156.857 ms

16  12.122.255.222  156.867 ms  156.641 ms  156.623 ms

17  216.148.209.66  159.982 ms  157.462 ms  158.537 ms

18  216.148.218.197  157.395 ms  156.789 ms  156.080 ms
```

Options for traceroute:

-f ttl change the initial time to live value to ttl instead of 1

-n do not resolve IP numbers

-v verbose

Module summary

Summary

The primary objective of this module was to give anoverview of the networking capabilities of the Linux operating system. Although one of the strengths of Linux is that plenty of information exists for nearly every component of it, most of this information is focused on implementation. New Linux users, particularly those coming from a Windows environment, are often unaware of the networking possibilities of Linux.

In this module you learned how to configure networking for your Linux box and to troubleshoot your Linux network connections a skill you will oftenly use as a system and network administrator within the Linux Environment

Assignment

Assignment

In the **Routing Scenario** section of this chapter give the routing table for the LAN's gateway.

2. Start your network interface manually

```
ifconfig eth0 192.168.0.x
```

List the kernel modules. Make sure that the eth0 module is loaded (check /etc/modules.conf).

3. Stop the network interface with:

(i) ifconfig eth0 down

Verify that you can bring the interface back up without entering new information:

(ii) ifconfig eth0 up

4. Stop the interface and remove the kernel module (rmmod *module*). What happens if you repeat step 3(ii)?

5. Divide the class into two networks A (192.168.1.0) and B (10.0.0.0).

- Try accessing machines across networks
  - Choose an existing machine to be the gateway (on either network)
  - **On the gateway machine only!** do the following:

-- allow IP forwarding:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

-- bring up an aliased interface (this will work as a second interface).

If you are on the 192.168.1.0 network then do the following

ifup eth0:1 10.0.0.*x* (where *x* is a an available IP).

add a route to the new network **forcing it to use the eth0:1 device**

-- add a route to the other network using the gateway machine (you will need to know either the eth0 or eth0:1 setting of this gw depending on which network you are on)

Module 16

Setting up Basic Networking Services: DNS, DHCP and LDAP

Introduction

Upon completion of this module you will be able to:

Outcomes

- Understand DNS, DHCP and Lightweight Directory Services (LDAP) plus be able to Install and Configure DNS, DHCP and LDAP

Terminology

Master Zone DNS:	The master zone includes all hosts from your network and a DNS server master zone stores up-to-date records for all the hosts in your domain.
Slave Zone DNS:	A slave zone is a copy of the master zone. The slave zone DNS server obtains its zone data with zone transfer operations from its master server. The slave zone DNS server responds authoritatively for the zone as long as it has valid The Domain Name System
DHCP	Dynamic Host Configuration Protocol (DHCP) is a network application protocol used by devices (DHCP clients) to obtain configuration information for operation in an Internet Protocol network. This protocol reduces system administration workload, allowing devices to be added to the network with little or no manual intervention.
LDAP	The Lightweight Directory Access Protocol, is an application protocol for querying and modifying directory services running over TCP/IP.A directory is a set of objects with attributes organized in a logical and hierarchical manner. A simple example is the telephone directory, which consists of a list of names (of either persons or organizations) organized alphabetically, with each name having an address and

phone number associated with it.

Domain Name Service

DNS assists in assigning an IP address to one or more names and assigning a name to an IP address. In Linux, this conversion is usually carried out by a special type of software known as bind. The machine that takes care of this conversion is called a name server. The names make up a hierarchical system in which each name component is separated by dots. The name hierarchy is, however, independent of the IP address hierarchy described above.

Consider a complete name, such as earth.example.com , written in the format hostname.domain. A full name, referred to as a fully qualified domain name (FQDN), consists of a hostname and a domain name (example.com). The latter also includes the top level domain or TLD (com). TLD assignment has become quite confusing for historical reasons. Traditionally, three letter domain names are used in the USA. In the rest of the world, the two-letter ISO national codes are the standard. In addition to that, longer TLDs were introduced in 2000 that represent certain spheres of activity (for example, .info, .name, .museum). In the early days of the Internet (before 1990), the file /etc/hosts was used to store the names of all the machines represented over the Internet. This quickly proved to be impractical in the face of the rapidly growing number of computers connected to the Internet. For this reason, a decentralized database was developed to store the hostnames in a widely distributed manner. This database, similar to the name server, does not have the data pertaining to all hosts in the Internet readily available, but can dispatch requests to other name servers.
The top of the hierarchy is occupied by root name servers. These root name servers manage the top level domains and are run by the Network Information Center (NIC). Each root name server knows about the name servers responsible for a given top level domain. Information about top level domain NICs is available at http://www.internic.net . DNS can do more than just resolve hostnames. The name server also knows which host is receiving e-mails for an entire domain—the mail exchanger (MX). For your machine to resolve an IP address, it must know about at least one name server and its IP address.

Setting up a DNS Server

When a program needs to resolve a host name it uses a mechanism called a resolver. The resolver will first consult the **/etc/nsswitch** file (previously **/etc/host.conf**) and determine which method should be used to resolve host names (local files, name server, NIS, or ldap server)

The /etc/host.conf (or /etc/nsswitch.conf) file

These files are scanned by the resolver. They indicate whether files, dns servers, ldap databases or nis servers should be consulted.

Example (/etc/nsswitch):

```
hosts:     files dns nis
networks:          files
```

The first line indicates that files (here /etc/hosts) should be queried first and then a DNS server if this fails. The second line instructs to use the /etc/network file for network information.

The /etc/hosts file

With a small number of networked computers it is possible to convert decimal IP numbers into names using the /etc/hosts file. The fields are as follows:

IP	machine	machine.domain alias

Example /etct/hosts file:

```
192.168.1.233    io            io.my.domain
61.20.187.42     callisto      callisto.physics.edu
```

The /etc/resolv.conf file

If the resolver needs to use a domain name server (DNS) then it will consult the **/etc/resolv.conf** file for a list of available servers to query from.

Hierarchical structure

Name servers have a hierarchical structure. Depending on the location in the fully qualified domain name (FQDM) a domain is called top-level, second-level or third-level.

Example of Top Level Domains

com	Commercial organisations
edu	US educational institutions
gov	US government institutions

mil	US military institutions
net	Gateways and network providers
org	Non commercial sites
uk	UK sites

Types of DNS servers

Domains can be further divided into sbdomains. This limits the amount of information needed to administer a domain. Zones have a **master** domain name server (previously called a **primary** DNS) and one or several **slave** domain name servers (previously called **secondary**). Administration of a name server consists of updating the information about a particular zone. The **master** servers are said to be authoritative.

DNS Configuration Files

In old versions of BIND (prior to BIND version 8) the configuration file was **/etc/named.boot**. With BIND version 8 the **/etc/named.conf** file is used instead. One can use the **named-bootconf.pl** utility to convert old configuration files.

The /etc/named.boot file:

```
directory                          /var/named
cache              .               named.ca
primary  myco.org                  named.myco
primary  0.0.127.in-addr.arp       named.local
primary  1.168.192.in-addr.arp     named.rev
```

The first line defines the base directory to be used. The name.ca file will contain a list of DNS IP addresses for querying external addresses. The third line is optional and contains records for the local LAN. The two next entries are for reverse lookups.

In **/etc/named.conf**:

cache	is replaced by *hint*
secondary	is replaced by *slave*
primary	is replaced by *master*.

Applying these changes to BIND4 configuration files will generate BIND8 and BIND9 files such as the following.

<u>The /etc/named.conf file</u>:

```
options {
                directory  "/var/named";
};
zone     "."        {
                type hint;
                file  "named.ca";
};
zone "myco.org"          {
                type master;
                file "named.myco";
};
zone "1.168.192.in-addr.arp" {
                type master;
                file "named.rev";
};
zone  "0.0.127.in-addr.arpa" {
                type master;
                file   "named.local";
};
```

<u>DNS zone files</u>

In this example the server is set as a caching-only server. All the zone files contain resource records.

<u>Sample **named.local** zone file:</u>

```
@      IN     SOA    localhost. root.localhost.  (
                               2001022700 ; Serial
                               28800      ; Refresh
                               14400      ; Retry
                               3600000    ; Expire
                               86400 )    ; Minimum
       IN     NS     localhost.
1      IN     PTR    localhost.
```

This is a very simple zone file but it gives us enough information to understand the basic mechanism of a name server.

The **@** sign will resolve to the related zone declared in **/etc/named.conf**. This allows any zone file to be used as a template for further zones (see the exercises).

Common Record Types

NS	Specify the zones primary name server
PTR	Reverse mapping of IP numbers to hostnames
MX	Mail exchange record
A	Associate an IP address with a hostname
CNAME	Associate an alias with the host's main name

Zone parameters

@ IN SOA	Start Of Authority. Identifies the zone followed by options enclosed in brackets.
serial	Is manually incremented when data is changed. Secondary servers query the master server's serial number. If it has changed, the entire zone file is downloaded

refresh	Time in seconds before the secondary server should query the SOA record of the primary domain. This should be at least a day.
retry	Time interval in seconds before attempting a new zone transfer if the previous download failed
expire	Time after which the secondary server discards all zone data if it contact the primary server. Should be a week at least
minimum	This is the ttl for the cached data. The default is one day (86400 seconds) but should be longer on stable LANs

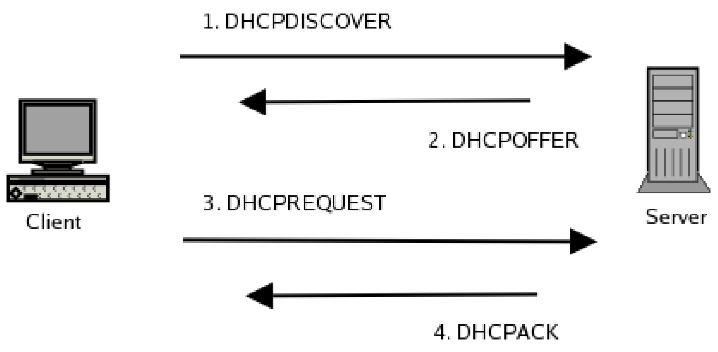
Reading

From the SuSE Linux Enterprise Server (Installation and Administration Document) read Pg 631 – 637 on DNS Implementation on SUSE Linux

Domain Host Configuration Protocol (DHCP)

WARNING!! You should not attempt to run a DHCP server unless you are certain not to interfere with the network you are currently using – The safest option for this section is to be totally isolated from the network and use a hub or a switch to connect the classroom together.

The basic communication process between a client workstation joining a TCP/IP network and the DHCP server is depicted below.



The DHCPDISCOVER request is sent using the broadcast 255.255.255.255 . The DHCP server can use two methods to allocate IP addresses:

1. A dynamic IP is assigned for a client host chosen from a range of IPs
2. A fixed IP is assigned for a specific host (identified using the MAC address, similar to bootp)

Since a single DHCP server can be used to administer IPs over several network, the **dhcpd.conf** configuration file is composed of global options followed by network sections:

Example network block:

```
subnet 10.0.0.0 netmask 255.0.0.0 {
....
}
```

In the next example we will assign both dynamic IP addresses and a fixed IP address:

```
subnet 10.0.0.0 netmask 255.0.0.0 {

        range 10.5.5.10 10.5.5.200;

        host  proxy {

                hardware ethernet 00:80:C6:30:0A:7E;

        fixed-address 10.5.5.2;

    }

}
```

For each subnet it is possible to give information on network services, such as

1. The default gateway
2. The DNS domain name and the NIS domain name
3. The DNS servers

In the subnet section above these directives would look like this:

```
        option routers              10.254.254.254;
        option nis-domain             "nisdomain";
        option domain-name             "seafront.bar";
        option domain-name-servers     10.0.0.2;
```

The database of dynamically assigned IP addresses is stored in **/var/lib/dhcp/dhcpd.leases**

Reading

From the SuSE Linux Enterprise Server (Installation and Administration Document) read Pg 637 – 653 on DHCP Implementation on SUSE Linux

LDAP

LDAP stands for Lightweight Directory Access Protocol. The protocol allows access to data in a tree-like structure using attributes. LDAP can be thought of as a specialised database which handles trees. Since directories are also trees, navigating LDAP fields is like navigating a directory. Added to this LDAP has been designed mainly for optimal access. This clarifies the words *Directory* and *Access*. With this in mind let's see what characterises an LDAP database.

The Distinguished Name

An item in the database can be referenced using a unique *Distinguished Name* (dn). This is similar to a file's full path in a directory. Each intermediate subfolder is called a *Relative Distinguished Name.*

Distinguished Name

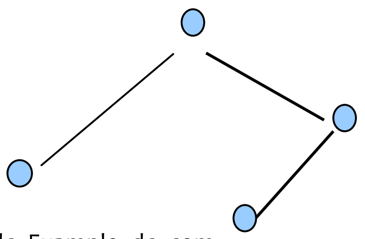
dc=example, dc=com

ou=People

ou=Aliases

cn=Tux

dn: cn=Tux, ou=People , dc=Example, dc=com



More Terminology

DIT	The Data Information Tree
DN	Distinguished Name
RDN	Relative Distinguished Name
LDIF	LDAP Data Interchange Format

Attributes:

dc	Domain Component
cn	Common Name
c	Country

l	Location
o	Organisation
ou	Organisational Module
sn	Surname
st	State
uid	User id

OpenLDAP server configuration

The server is called **slapd** (Standalone LDAP daemon) and it's configuration file is:

/etc/openldap/slapd.conf

We will cover each section of this file in more detail

Importing schemas

There is an *include* clause in **slapd.conf** which tells the LDAP server which schemas should be loaded.

We need at least the following:

```
Include      /etc/openldap/schema/core.schema
include      /etc/openldap/schema/misc.schema
include      /etc/openldap/schema/cosine.schema
include      /etc/openldap/schema/nis.schema
include      /etc/openldap/schema/inetorgperson.schema
```

Database Definition

Available DBMs (Database Managers) are *ldbm* or the more recent *bdb*. We will use bdb:

```
database     bdb
```

You need to specify the root or base for the LDAP directory, as well as the directory where the database file will be kept. This is done below;

```
suffix       “dc=example,dc=com”
```

```
directory          /var/lib/ldap/
```

The following lines are only needed when modifying the LDAP server online. You can then specify an adminstrator username/password. Use the **slappasswd** to generate an encrypted hash:

```
rootdn       "cn=Manager,dc=example,dc=com"
rootpw       {SSHA}KiXS5htbnVEQp7OrjoteQZHHICs0krBO
```

Client configuration files

There are two configuration files called ldap.conf. Here is what they do:

- The /etc/ldap.conf file is used by the nss_ldap and pam_ldap modules
- The file /etc/openldap/ldap.conf is used by the tools **ldapsearch** and **ldapadd**

For example, to save time typing:

```
ldapsearch -b "dc=example,dc=com"  -x
```

you can add the next lines to **/etc/openldap/ldap.conf**

```
BASE     dc=example, dc=com
HOST               127.0.0.1
```

So far we have configured **slapd** and the configuration file for **ldapsearch** in particular. Once we have populated an LDAP directory we will be able to test our setup by typing:

```
ldapsearch -x
```

Migrating System Files to LDAP

There are two methods available to populate an LDAP directory.

- If the ldap daemon **slapd** is stopped, we can do an *offline* update using **slapadd**
- While **slapd** is running, it is possible to perform an *online* update using **ldapadd** or **ldapmodify**

We will also use migration tools which can be downloaded from:

http://www.padl.com/OSS/MigrationTools.html

Creating LDAP directories *offline*

We are going to work in the directory containing the LDAP migration Perl scripts which we have downloaded from www.padl.com.

Notice: Some distributions may include the migration tools with the LDAP server package.

You should have the following files:

migrate_automount.pl	migrate_base.pl
CVSVersionInfo.txt	migrate_common.ph
Make.rules	migrate_fstab.pl
MigrationTools.spec	migrate_group.pl
README	migrate_hosts.pl
ads	migrate_netgroup.pl
migrate_netgroup_byhost.pl	migrate_aliases.pl
migrate_netgroup_byuser.pl	migrate_all_netinfo_offline.sh
migrate_networks.pl	migrate_all_netinfo_online.sh
migrate_passwd.pl	migrate_all_nis_offline.sh
migrate_profile.pl	migrate_all_nis_online.sh
migrate_protocols.pl	migrate_all_nisplus_offline.sh
migrate_rpc.pl	migrate_all_nisplus_online.sh
migrate_services.pl	migrate_all_offline.sh
migrate_slapd_conf.pl	migrate_all_online.sh

First edit **migrate_common.ph** and change the $DEFAULT_BASE variable to:

$DEFAULT_BASE = "dc=example,dc=com";

NOTICE

When migrating the /etc/passwd file one can either use shadow passwords or not. When using shadow passwords an added objectClass called shadowAccount is used in the LDAP record and there is no need to migrate the shadow password file.

We create our first LDIF file called **base.ldif** to serve as our root:

```
/migrate_base.pl > base.ldif
```

This flat file will be converted into bdb (or ldbm) files stored in **/var/lib/ldap** as follows:

```
slapadd -v < base.ldif
```

We next choose to migrate the password without shadow passwords as follows:

```
pwunconv
```

```
./migrate_passwd.pl /etc/passwd passwd.ldif
```

The entries in **passwd.ldif** should look like this:

dn: uid=test,ou=People,dc=example,dc=com

uid: test

cn: test

objectClass: account

objectClass: posixAccount

objectClass: top

userPassword: {crypt}$1$FGrRfa0u$lo5XwA9xxssmjb0NB2Z361

loginShell: /bin/bash

uidNumber: 505

gidNumber: 506

homeDirectory: /home/test

Now let's add this LDIF file to our LDAP directory:(remember that LDAP is stopped so we are still offline)

```
slapadd -v -l passwd.ldif  or
slapadd -v < passwd.ldif
```

NOTICE:
Make sure all the files in **/var/lib/ldap** belong to user **ldap**

TESTING:

Restart the LDAP server

```
/etc/init.d/ldap restart
```

Search all the entries in the directory:

```
ldapsearch -x
```

If the **ldap** server does not respond, or the result from **ldapsearch** is empty, it is possible to show the content of the LDAP databases in **/var/lib/ldap** with the **slapcat** command.

Creating LDAP Directories Online

The LDAP server can be updated online, without having to shut the ldap service down. For this to work however we must specify a **rootdn** and a **rootpw** in **/etc/openldap/slapd.conf**.

The password is generated from the command line as follows

```
sldappasswd

New password:

Re-enter new password:

{SSHA}XyZmHH1RlnSVXTj87UvxOAOCZA8oxNCT
```

We next choose the **rootdn** in **/etc/openldap/slapd.conf** to be

```
rootdn      "cn=Manager,dc=example,dc=com"
rootpw      {SSHA}XyZmHH1RlnSVXTj87UvxOAOCZA8oxNCT
```

The next line will update the LDAP entries

```
ldapmodify -f passwd.ldif -x -D “dc=example,dc=com” -W

Enter LDAP Password:
```

LDAP Authentication Scheme

Server Configuration

We assume that the LDAP server has been configured as above.

The passwords in the LDAP directory can also be updated online with the **ldappasswd** command.

The next line will update the password for user *tux* on the LDAP server.

```
ldappasswd    -D   "cn=Manager,dc=example,dc=com"    -S   -x   -W    \
"uid=tux,ou=People,dc=example,dc=com"
```

The **-S** switch is used to configure a new password.

We assume that the IP address for the server is 10.0.0.1 and that the domain component is “dc=example,dc=com”

You may allow users to change their passwords on the LDAP server as follows:

1. Copy the *passwd* PAM file **/etc/share/doc/nss_ldap**-*version*/**pam.d/passwd** to **/etc/pam.d**

2. Add the following access rule in **/etc/openldap/slapd.conf**

```
access to attrs=userPassword
 by self write
 by anonymous auth
 by * none
```

Client Configuration

The clients need to have the **nss_ldap** package installed (some distributions have a separate **pam_ldap** package with the PAM related modules and files). The following files and libraries are installed:

/etc/ldap.conf	set the hostname and the domain component of the LDAP server used for authentications
/lib/libnss_ldap-2.3.2.so	an ldap module for the NameService Switch
/lib/security/pam_ldap.so	the PAM ldap module
/usr/lib/libnss_ldap.so	a symbolic link to /lib/libnss_ldap-2.3.2.so
/usr/share/doc/nss_ldap-207/pam.d	sample files for programs using PAM

If we don't use SSL certificates then **/etc/ldap.conf** is as follows:

The **/etc/ldap.conf** file

```
host 10.0.0.1
base dc=example,dc=com
ssl no
pam_password md5
```

Next in **/etc/pam.d** replace the file called **login** with **/usr/share/doc/nss_ldap-207/pam.d/login**. This will tell the authentication binary **/bin/login** to use the pam_ldap.so module.

Finally the **/etc/nsswitch.conf** needs to have the following line:

```
passwd ldap files
```

Check the **/var/log/ldap/ldap.log** file on the server to follow the authentication process.

Reflection

Compare Linux Open LDAP Directory, DHCP and DNS Implementation with Windows

Submit your comments and observations on your Wiki Page inside the http://www.colwiki.org/Reflections_and_Assignments webpage

Reading

From the SuSE Linux Enterprise Server (Installation and Administration Document) read Pg 663 – 697 on LDAP Implementation on SUSE Linux

Module summary

Summary

In this module you learned some of the backbone services required of Linux in any organization.The LDAP Service is critical where you would like to maintain a central point of authentication for all your users in the system. DNS and DHCP Services are very critical for the management of the Local Area Network.

Assignment

Assignment

You should by now have installed webmin (www.webmin.com), you are required to use Webmin interface to configure a working DNS, DHCP and LDAP server. Note the webmin options that you will need to specify or change.

Post your results to this manual website in www.colwiki.org

Module 17

Web/Internet Gateway

Introduction

This module will enable you to understand Network Proxy Servers, Install and configure the Squid Proxy Server and:

Outcomes

- Configure the Squid Proxy Server to offer controlled, authenticated and validated internet gateway access
- Installation and Configuration of APACHE Webserver
- Advanced Apache Web server configuration and integration to Databases
- Apache Web server Performance tuning

Terminology

Proxy Server:	A proxy server is a server (a computer system or an application program) that acts as a go-between for requests from clients seeking resources from other servers. A client connects to the proxy server, requesting some service, such as a file, connection, web page, or other resource, available from a different server. The proxy server evaluates the request according to its filtering rules
Web Server:	A computer program that is responsible for accepting HTTP requests from clients (user agents such as web browsers), and serving them HTTP responses along with optional data contents, which usually are web pages such as HTML documents and linked objects (images, etc.).

Web Services

There are a number of Web Servers that can be used in the Linux platform, however in this manual, we will use the apache webserver also referred to as HTTPD. With a share of more than 70%, the Apache HTTP Server (Apache) is the world's most widely-used Web server according to the Survey from http://www.netcraft.com/ . Apache, developed by the Apache Software Foundation (http://www.apache.org/), is available for most operating systems.

You can install apache from the rpms or debian files that come with most of the distribution CDs if it is not already installed by default. You can use any of the automated installers that come with the distribution that you are using.

Apache is controlled by a series of configuration files: **httpd.conf**, **access.conf**. and **srm.conf** (there's actually also a **mime.types** file, but you have to deal with that only when you're adding or removing MIME types from your server, which shouldn't be too often). The files contain instructions, called directives that tell Apache how to run. Several companies offer GUI-based Apache front-ends, but it's easier to edit the configuration files by hand.

Use the chkconfig command to configure Apache to start at boot:

```
[root@linux]# chkconfig httpd on
```

Use the httpd<code> init script in the <code>/etc/init.d directory to start,stop, and restart Apache after booting:

```
[root@linux]# /etc/init.d/httpd start

[root@linux]# /etc/init.d/httpd stop

[root@linux]# /etc/init.d/httpd restart
```

You can test whether the Apache process is running with

```
[root@linux]# pgrep httpd
```

you should get a response of plain old process ID numbers.

This process may differ between distributions. When using Debian based distribution packages you will note that instead of the httpd you will trype the word apache or apache2.

The configuration file used by Apache is /etc/httpd/conf/httpd.conf in Redhat / Fedora distributions and /etc/apache*/httpd.conf in Debian / Ubuntu distributions. As for most Linux applications, you must restart Apache before changes to this configuration file take effect.

All the statements that define the features of each web site are grouped together inside their own <VirtualHost> section, or container, in the httpd.conf file. The most commonly used statements, or directives, inside a <VirtualHost> container are:

* servername: Defines the name of the website managed by the <VirtualHost> container. This is needed in named virtual hosting only, as I'll explain soon.

* DocumentRoot: Defines the directory in which the web pages for the site can be found.

By default, Apache searches the DocumentRoot directory for an index, or home, page named index.html. So for example, if you have a servername of www.my-site.com with a DocumentRoot directory of /home/www/site1/, Apache displays the contents of the file /home/www/site1/index.html when you enter http://www.my-site.com in your browser.

Some editors, such as Microsoft FrontPage, create files with an .htm extension, not .html. This isn't usually a problem if all your HTML files have hyperlinks pointing to files ending in .htm as FrontPage does. The problem occurs with Apache not recognizing the topmost index.htm page. The easiest solution is to create a symbolic link (known as a shortcut to Windows users) called index.html pointing to the file index.htm. This then enables you to edit or copy the file index.htm with index.html being updated automatically. You'll almost never have to worry about index.html and Apache again!

This example creates a symbolic link to index.html in the /home/www/site1 directory.

```
[root@linux]# cd /home/www/site1

[root@linux]# ln -s index.htm index.html

[root@linux]# ll index.*

-rw-rw-r--   1 root    root      48590 Jun 18 23:43 index.htm

lrwxrwxrwx   1 root    root          9 Jun 21 18:05 index.html -> index.htm

[root@linux]#
```

The l at the very beginning of the index.html entry signifies a link and the -> the link target.

The Default File Location

By default, Apache expects to find all its web page files in the /var/www/html/ directory with a generic DocumentRoot statement at the beginning of httpd.conf. The examples in this chapter use the /home/www directory to illustrate how you can place them in other locations successfully.

File Permissions and Apache

Apache will display Web page files as long as they are world readable. You have to make sure you make all the files and subdirectories in your DocumentRoot have the correct permissions.

It is a good idea to have the files owned by a nonprivileged user so that Web developers can update the files using FTP or SCP without requiring the root password.

To do this:

1. Create a user with a home directory of /home/www.

2. Recursively change the file ownership permissions of the /home/www directory and all its subdirectories.

3. Change the permissions on the /home/www directory to 755, which allows all users, including the Apache's httpd daemon, to read the files inside.

```
[root@linux]# useradd -g users www

[root@linux]# chown -R www:users /home/www

[root@linux]# chmod 755 /home/www
```

Now we test for the new ownership with the ll command.

```
[root@linux]# ll /home/www/site1/index.*

-rw-rw-r--   1 www    users     48590 Jun 25 23:43 index.htm

lrwxrwxrwx   1 www    users         9 Jun 25 18:05 index.html -> index.htm

[root@linux]#
```

Note: Be sure to FTP or SCP new files to your web server as this new user. This will make all the transferred files automatically have the correct ownership.

If you browse your Web site after configuring Apache and get a "403 Forbidden" permissions-related error on your screen, then your files or directories under your DocumentRoot most likely have incorrect permissions. You may also have to use the Directory directive to make Apache serve the pages once the file permissions have been correctly set. If you have your files in the default /home/www directory then this second step becomes unnecessary.

Named Virtual Hosting

You can make your Web server host more than one site per IP address by using Apache's named virtual hosting feature. You use the NameVirtualHost directive in the /etc/httpd/conf/httpd.conf file to tell Apache which IP addresses will participate in this feature.

The <VirtualHost> containers in the file then tell Apache where it should look for the Web pages used on each Web site. You must specify the IP address for which each <VirtualHost> container applies.

Named Virtual Hosting Example

Consider an example in which the server is configured to provide content on 97.158.253.26. In the code that follows, notice that within each <VirtualHost> container you specify the primary Web site domain name for that IP address with the ServerName directive. The DocumentRoot directive defines the directory that contains the index page for that site.

You can also list secondary domain names that will serve the same content as the primary ServerName using the ServerAlias directive.

Apache searches for a perfect match of NameVirtualHost, <VirtualHost>, and ServerName when making a decision as to which content to send to the remote user's Web browser. If there is no match, then Apache uses the first <VirtualHost> in the list that matches the target IP address of the request.

This is why the first <VirtualHost> statement contains an asterisk: to indicate it should be used for all other Web queries.

```
NameVirtualHost 97.158.253.26
<VirtualHost *>
   Default Directives. (In other words, not site
#1 or site #2)
</VirtualHost>
<VirtualHost 97.158.253.26>
   servername www.my-site.com
   Directives for site #1
</VirtualHost>
<VirtualHost 97.158.253.26>
   servername www.another-site.com
   Directives for site #2
</VirtualHost>
```

Be careful with using the asterisk in other containers. A <VirtualHost> with a specific IP address always gets higher priority than a <VirtualHost> statement with an * intended to cover the same IP address, even if the ServerName directive doesn't match. To get consistent results, try to limit the use of your <VirtualHost *> statements to the beginning of the list to cover any other IP addresses your server may have.

You can also have multiple NameVirtualHost directives, each with a single IP address, in cases where your Web server has more than one IP address.

IP-Based Virtual Hosting

The other virtual hosting option is to have one IP address per Web site, which is also known as IP-based virtual hosting. In this case, you will not have a NameVirtualHost directive for the IP address, and you must only have a single <VirtualHost> container per IP address.

Also, because there is only one Web site per IP address, the ServerName directive isn't needed in each <VirtualHost> container, unlike in named virtual hosting.

IP Virtual Hosting Example: Single Wild Card

In this example, Apache listens on all interfaces, but gives the same content. Apache displays the content in the first <VirtualHost *> directive even if you add another right after it. Apache also seems to enforce the single <VirtualHost> container per IP address requirement by ignoring any ServerName directives you may use inside it.

```
<VirtualHost *>

   DocumentRoot /home/www/site1

</VirtualHost>
```

IP Virtual Hosting Example: Wild Card and IP addresses

In this example, Apache listens on all interfaces, but gives different content for addresses 97.158.253.26 and 97.158.253.27. Web surfers get the site1 content if they try to access the web server on any of its other IP addresses.

```
<VirtualHost *>
   DocumentRoot /home/www/site1
</VirtualHost>
<VirtualHost 97.158.253.26>
   DocumentRoot /home/www/site2
</VirtualHost>
<VirtualHost 97.158.253.27>
   DocumentRoot /home/www/site3
</VirtualHost>
```

`Because it makes configuration` easier, system administrators commonly replace the IP address in the <VirtualHost> and NameVirtualHost directives with the * wildcard character to indicate all IP addresses.

If you installed Apache with support for secure HTTPS/SSL, which is used frequently in credit card and shopping cart Web pages, then wild cards won't work. The Apache SSL module demands at least one explicit <VirtualHost> directive for IP-based virtual hosting. When you use wild cards, Apache interprets it as an overlap of name-based and IP-based <VirtualHost> directives and gives error messages because it can't make up its mind about which method to use:

Starting httpd: [Sat Oct 12 21:21:49 2002] [error] VirtualHost _default_:443 -- mixing * ports and non-* ports with a NameVirtualHost address is not supported, proceeding with undefined results

If you try to load any Web page on your web server, you'll see the error:

Configuration - Multiple Sites And IP Addresses

To help you better understand the edits needed to configure the /etc/httpd/conf/httpd.conf file, I'll walk you through an example scenario. The parameters are:

1. The web site's systems administrator previously created DNS entries for www.my-site.com, my-site.com, www.my-cool-site.com and www.default-site.com to map the IP address 97.158.253.26 on this web server. The domain www.another-site.com is also configured to point to alias IP address 97.158.253.27. The administrator wants to be able to get to www.test-site.com on all the IP addresses.

2. Traffic to www.my-site.com, my-site.com, and www.my-cool-site.com must get content from subdirectory site2. Hitting these URLs causes Apache to display the contents of file index.html in this directory.

3. Traffic to www.test-site.com must get content from subdirectory site3.

4. Named virtual hosting will be required for 97.158.253.26 as in this case we have a single IP address serving different content for a variety of domains. A NameVirtualHost directive for 97.158.253.26 is therefore required.

5. Traffic going to www.another-site.com will get content from directory site4.

6. All other domains pointing to this server that don't have a matching ServerName directive will get Web pages from the directory defined in the very first <VirtualHost> container: directory site1. Site www.default-site.com falls in this category.

Securing a Webserver using HTTPS

The secure socket layer protocol SSL allows any networked applications to use encryption. This can be thought of as a process which wraps the socket preparing it to use encryption at the application level. In the case of HTTPS, the server uses a pair of keys, public and private. The server's public key is used by the client to encrypt the session key, the private key is then used to decrypt the session key for use.

The public key is published using certificates. A certificate contains the following information:

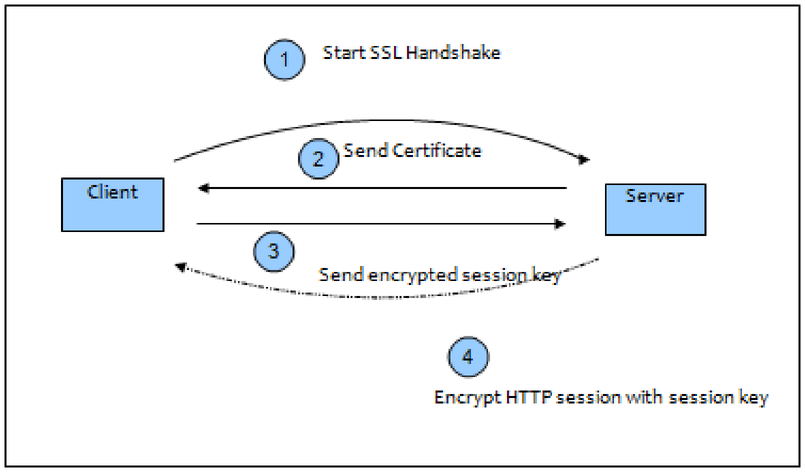
- Name and Address, Hostname, etc.

- Public Key

- TTL

- (optional) ID + Signature from a certificate authority (CA)

The certificate will be used to establish the authenticity of the server. A valid signature from a known CA is automatically recognised by the client's browser. With Mozilla for example these trusted CA certificates can be found by following the links: **Edit -> Preferences -> Privacy & Security -> Certificates** then clicking on the "*Manage Certificates*" button and the Authorities TAB


1 Start SSL Handshake
2 Send Certificate
Client
Server
3 Send encrypted session key
4 Encrypt HTTP session with session key

On the other hand communications would be too slow if the session was encrypted using public key encryption. Instead, once the authenticity of the server is established, the client generates a unique secret session key which is encrypted using the servers public key found in the certificate. Once the server receives this session key it can decrypt it using the private key associated with the certificate. From there on the communication is encrypted and decrypted using this secrete session key generated by the client.

SSL Virtual Hosts

A separate apache server can be used to listen on port 443 and implement SSL connections. However most default configurations involve a single apache server listening on both ports 80 and 443.

For this an additional **Listen** directive is set in **httpd.conf** asking the server to listen on port 443. Apache will then bind to both ports 443 and 80. Non encrypted connections are handled on port 80 while an SSL aware virtual host is configured to listen on port 443:

```
<VirtualHost _default_:443>
      SSL CONFIGURATION
</VirtualHost>
```

The SSL CONFIGURATION lines are:

```
SSLEngine on

SSLCipherSuite
ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+S
SLv2:+EXP

SSLCertificateFile PATH_TO_FILE.crt

SSLCertificateKeyFile PATH_TO_FILE.key
```

We need to generate the servers private key (FILE.key) and certificate (FILE.crt) to complete this configuration.

Managing Certificates

The keys and certificates are usually kept in subdirectories of **/etc/httpd/conf** called **ssl.crt** and **ssl.key**.

There should also be a Makefile that will generate both a KEY and a CERTIFICATE in PEM format which is base64 encoded data.

Using the Makefile

For example if we want to generate a self-signed certificate and private key simply type:

```
make mysite.crt
```

The Makefile will generate both files mysite.key (the private key) as well as mysite.crt (the certificate file containing the public key). You can use the following directives in **httpd.conf**:

SSLCertificateFile ... *mysite.crt*

SSLCertificateKeyFile ... *mysite.key*

Certificate Requests

On a production server you would need to generate a new file called a "certificate request" with:

```
openssl  req -new  -key mysite.key -out mysite.csr
```

This file can be sent to a certificate authority (CA) to be signed. The certificate authority will send back the signed certificate.

Pass Phrases

A private key can be generated with or without a passphase, and a private key without a passphrase can be constructed from an existing private key.

A passphrased file: If a private key has a passphrase set then the file starts with

```
-----BEGIN RSA PRIVATE KEY-----

Proc-Type: 4,ENCRYPTED

DEK-Info: DES-EDE3-CBC, ---- snip ----

.....
```

this means that the file is protected by a pass-phrase using 3DES. This was generate by the line

/usr/bin/openssl genrsa **-des3** 1024 > $@ in the Makefile. If the -des3 flag is omitted NO passphrase is set. You can generate a new private key (mysite-nophrase.key) without a passphrase from the old private key (mysite.key) as follows:

```
openssl rsa -in mysite.key -out mysite-nopass.key
```

Reading

From the SuSE Linux Enterprise Server (Installation and Administration Document) read Pg 741 – 778 on APACHE Implementation on SUSE Linux.

Internet Gateway

An internet gateway is useful in two ways:

- Reduce Internet bandwidth charges
- Limit access to the Web to only authorized users.

In linux this is achieved by the Squid Proxy Server. The Squid web caching proxy server can achieve both these goals fairly easily. Users configure their web browsers to use the Squid proxy server instead of going to the web directly. The Squid server then checks its web cache for the web information requested by the user. It will return any matching information that finds in its cache, and if not, it will go to the web to find it on behalf of the user. Once it finds the information, it will populate its cache with it and also forward it to the user's web browser.

This reduces the amount of data accessed from the web. Another advantage is that you can configure your firewall to only accept HTTP web traffic from the Squid server and no one else. Squid can then be configured to request usernames and passwords for each user that users its services. This provides simple access control to the Internet.

Squid can be installed through the relevant packages of Package application managers provided by the relevant distribution. To start squid on boot use the chkconfig.

```
[root@linux]# chkconfig squid on
```

Use the service command to start, stop, and restart Squid after booting:

```
[root@linux]# service squid start
[root@linux]# service squid stop
[root@linux]# service squid restart
```

You can test whether the Squid process is running with the pgrep command:

```
[root@linux]# pgrep squid
```

You should get a response of plain old process ID numbers. The /etc/squid/squid.conf File. The main Squid configuration file is squid.conf, and, like most Linux applications, Squid needs to be restarted for changes to the configuration file can take effect.

The Visible Host Name

Squid will fail to start if you don't give your server a hostname. You can set this with the visible_hostname parameter. Here, the hostname is set to the real name of the server bigboy.

```
visible_hostname linux
```

Access Control Lists

You can limit users' ability to browse the Internet with access control lists (ACLs). Each ACL line defines a particular type of activity, such as an access time or source network, they are then linked to an http_access statement that tells Squid whether or not to deny or allow traffic that matches the ACL.

Squid matches each Web access request it receives by checking the http_access list from top to bottom. If it finds a match, it enforces the allow or deny statement and stops reading further. You have to be careful not to place a deny statement in the list that blocks a similar allow statement below it. The final http_access statement denies everything, so it is best to place new http_access statements above it

Note: The very last http_access statement in the squid.conf file denies all access. You therefore have to add your specific permit statements above this line. In the chapter's examples, I've suggested that you place your statements at the top of the http_access list for the sake of manageability, but you can put them anywhere in the section above that last line.

Squid has a minimum required set of ACL statements in the ACCESS_CONTROL section of the squid.conf file. It is best to put new customized entries right after this list to make the file easier to read.
Restricting Web Access By Time

You can create access control lists with time parameters. For example, you can allow only business hour access from the home network, while always restricting access to host 192.168.1.23.

```
#
# Add this to the bottom of the ACL section of squid.conf
#
acl home_network src 192.168.1.0/24
acl business_hours time M T W H F 9:00-17:00
acl RestrictedHost src 192.168.1.23

#
# Add this at the top of the http_access section of squid.conf
#
http_access deny RestrictedHost
http_access allow home_network business_hours

Or, you can allow morning access only:

#
# Add this to the bottom of the ACL section of squid.conf
#
acl mornings time 08:00-12:00

#
# Add this at the top of the http_access section of squid.conf
#
http_access allow mornings
```

Restricting Access to specific Web sites

Squid is also capable of reading files containing lists of web sites and/or domains for use in ACLs. In this example we create to lists in files named /usr/local/etc/allowed-sites.squid and /usr/local/etc/restricted-sites.squid.

```
# File: /usr/local/etc/allowed-sites.squid
www.col.org
www.yahoo.com

# File: /usr/local/etc/restricted-sites.squid
www.porn.com
illegal.com
```

These can then be used to always block the restricted sites and permit the allowed sites during working hours. This can be illustrated by expanding our previous example slightly.

```
#
# Add this to the bottom of the ACL section of squid.conf
#
acl home_network src 192.168.1.0/24
acl business_hours time M T W H F 9:00-17:00
acl GoodSites dstdomain "/usr/local/etc/allowed-sites.squid"
acl    BadSites       dstdomain    "/usr/local/etc/restricted-
sites.squid"

#
#  Add  this  at  the  top  of  the  http_access  section  of
squid.conf
#
http_access deny BadSites
http_access allow home_network business_hours GoodSites
```

Restricting Web Access By IP Address

You can create an access control list that restricts Web access to users on certain networks. In this case, it's an ACL that defines a home network of 192.168.1.0.

```
#
# Add this to the bottom of the ACL section of squid.conf
#
acl home_network src 192.168.1.0/255.255.255.0
```

You also have to add a corresponding http_access statement that allows traffic that matches the ACL:

```
#
#  Add  this  at  the  top  of  the  http_access  section  of
squid.conf
#
http_access allow home_network
```

Password Authentication Using NCSA

You can configure Squid to prompt users for a username and password. Squid comes with a program called ncsa_auth that reads any NCSA-compliant encrypted password file. You can use the htpasswd program that comes installed with Apache to create your passwords. Here is how it's done:

1) Create the password file. The name of the password file should be /etc/squid/squid_passwd, and you need to make sure that it's universally readable.

```
[root@linux]# touch /etc/squid/squid_passwd
[root@linux]# chmod o+r /etc/squid/squid_passwd
```

2) Use the htpasswd program to add users to the password file. You can add users at anytime without having to restart Squid. In this case, you add a username called www:

```
[root@linux]# htpasswd /etc/squid/squid_passwd www
New password:
Re-type new password:
```

```
Adding password for user www
[root@linux]#
```

3) Find your ncsa_auth file using the locate command.

```
[root@linux]# locate ncsa_auth
/usr/lib/squid/ncsa_auth
[root@linux]#
```

4) Edit squid.conf; specifically, you need to define the authentication program in squid.conf, which is in this case ncsa_auth. Next, create an ACL named ncsa_users with the REQUIRED keyword that forces Squid to use the NCSA auth_param method you defined previously. Finally, create an http_access entry that allows traffic that matches the ncsa_users ACL entry. Here's a simple user authentication example; the order of the statements is important:

```
#
# Add this to the auth_param section of squid.conf
#
auth_param     basic     program     /usr/lib/squid/ncsa_auth
/etc/squid/squid_passwd

#
# Add this to the bottom of the ACL section of squid.conf
#
acl ncsa_users proxy_auth REQUIRED

#
#  Add  this  at  the  top  of  the  http_access  section  of
squid.conf
#
http_access allow ncsa_users
```

5) This requires password authentication and allows access only during business hours. Once again, the order of the statements is important:

```
#
# Add this to the auth_param section of squid.conf
#
auth_param     basic     program     /usr/lib/squid/ncsa_auth
/etc/squid/squid_passwd

#
# Add this to the bottom of the ACL section of squid.conf
#
acl ncsa_users proxy_auth REQUIRED
acl business_hours time M T W H F 9:00-17:00

#
#  Add  this  at  the  top  of  the  http_access  section  of
squid.conf
#
http_access allow ncsa_users business_hours
```

Remember to restart Squid for the changes to take effect.

Forcing Users To Use Your Squid Server

If you are using access controls on Squid, you may also want to configure your firewall to allow only HTTP Internet access to only the Squid server. This forces your users to browse the Web through the Squid proxy.
Making Your Squid Server Transparent To Users

It is possible to limit HTTP Internet access to only the Squid server without having to modify the browser settings on your client PCs. This called a transparent proxy configuration. It is usually achieved by configuring a firewall between the client PCs and the Internet to redirect all HTTP (TCP port 80) traffic to the Squid server on TCP port 3128, which is the Squid server's default TCP port.

Squid Transparent Proxy Configuration

Your first step will be to modify your squid.conf to create a transparent proxy. The procedure is different depending on your version of Squid.

Prior to version 2.6: In older versions of Squid, transparent proxy was achieved through the use of the httpd_accel options which were originally developed for http acceleration. In these cases, the configuration syntax would be as follows:

```
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

Version 2.6 and Beyond: Newer versions of Squid simply require you to add the word "transparent" to the default "http_port 3128" statement. In this example, Squid not only listens on TCP port 3128 for proxy connections, but will also do so in transparent mode.

```
http_port 3128 transparent
```

Configuring iptables to Support the Squid Transparent Proxy

The examples below are based on the discussion of Linux iptables in Chapter 14, "Linux Firewalls Using iptables". Additional commands may be necessary for you particular network topology.

In both cases below, the firewall is connected to the Internet on interface eth0 and to the home network on interface eth1. The firewall is also the default gateway for the home network and handles network address translation on all the network's traffic to the Internet.

Only the Squid server has access to the Internet on port 80 (HTTP), because all HTTP traffic, except that coming from the Squid server, is redirected.

If the Squid server and firewall are the same server, all HTTP traffic from the home network is redirected to the firewall itself on the Squid port of 3128 and then only the firewall itself is allowed to access the Internet on port 80.

```
        iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 \
                -j REDIRECT --to-port 3128
        iptables -A INPUT -j ACCEPT -m state \
                --state NEW,ESTABLISHED,RELATED -i eth1 -p tcp \
        --dport 3128
iptables -A OUTPUT -j ACCEPT -m state \
        --state NEW,ESTABLISHED,RELATED -o eth0 -p tcp \
        --dport 80
iptables -A INPUT -j ACCEPT -m state \
        --state ESTABLISHED,RELATED -i eth0 -p tcp \
        --sport 80
iptables -A OUTPUT -j ACCEPT -m state \
        --state ESTABLISHED,RELATED -o eth1 -p tcp \
        --sport 80
```

Note: This example is specific to HTTP traffic. You won't be able to adapt this example to support HTTPS web browsing on TCP port 443, as that protocol specifically doesn't allow the insertion of a "man in the middle" server for security purposes. One solution is to add IP masquerading statements for port 443, or any other important traffic, immediately after the code snippet. This will allow non HTTP traffic to access the Internet without being cached by Squid.

If the Squid server and firewall are different servers, the statements are different. You need to set up iptables so that all connections to the Web, not originating from the Squid server, are actually converted into three connections; one from the Web browser client to the firewall and another from the firewall to the Squid server, which triggers the Squid server to make its own connection to the Web to service the request. The Squid server then gets the data and replies to the firewall which then relays this information to the Web browser client. The iptables program does all this using these NAT statements:

```
iptables -t nat -A PREROUTING -i eth1 -s ! 192.168.1.100 \
        -p tcp --dport 80 -j DNAT --to 192.168.1.100:3128
iptables -t nat -A POSTROUTING -o eth1 -s 192.168.1.0/24 \
        -d 192.168.1.100 -j SNAT --to 192.168.1.1
iptables -A FORWARD -s 192.168.1.0/24 -d 192.168.1.100 \
        -i eth1 -o eth1 -m state
         --state NEW,ESTABLISHED,RELATED \
        -p tcp --dport 3128 -j ACCEPT
 iptables -A FORWARD -d 192.168.1.0/24 -s 192.168.1.100 \
        -i eth1 -o eth1 -m state --state ESTABLISHED,RELATED \
        -p tcp --sport 3128 -j ACCEPT
```

In the first statement all HTTP traffic from the home network except from the Squid server at IP address 192.168.1.100 is redirected to the Squid server on port 3128 using destination NAT. The second statement makes this redirected traffic also undergo source NAT to make it appear as if it is coming from the firewall itself. The FORWARD statements are used to ensure the traffic is allowed to flow to the Squid server after the NAT process is complete. The unusual feature is that the NAT all takes place on one interface; that of the home network (eth1).

Apache Log File Analysis Software

Most log analysis tools available for squid are listed on the following site:

http://www.squid-cache.org/Scripts/

The main logfile for squid is the **/var/log/squid/access.log** file. Next is a short overview of **calamaris** and **webalizer**. Also notice that **webmin** produces log reports based on calamaris.

- **Calamaris**

The code is GPL and can be downloaded from http://cord.de/tools/squid/calamaris. You can generate reports as follow:

```
cat /var/log/squid/access.log | calamaris
```

In order to get information on webpage requests per host one can use the **-R** switch: There are many more switches available (check the manpages for calamaris). There are also a number of scripts that can run hourly or monthly reports. These scipts are included in the EXAMPLES file distributed with calamaris.

```
calamaris -R 5 /var/log/squid/access.log
```

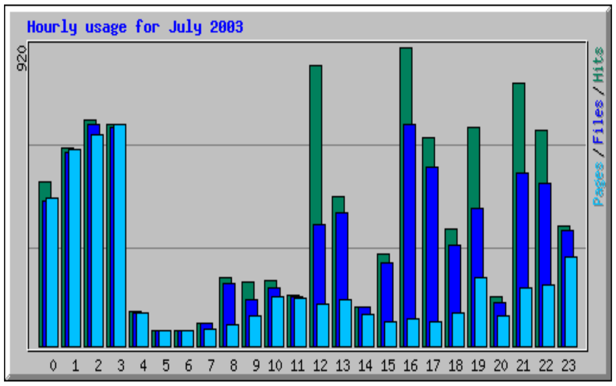
- # Incoming TCP-requests by host

```
host / target                     request   hit-%   Byte    hit-% sec   kB/sec
--------------------------------- --------- ------ -------- ------ ---- -------
192.168.2.103                       72    0.00   323336   0.00    0   10.24
 *.redhat.com                       35    0.00   126726   0.00    0   10.44
 *.suse.co.uk                       20    0.00    63503    0.00    0   13.15
 *.lemonde.fr              6      0.00   109712   0.00    1   16.39
 207.36.15.*               5      0.00     8946     0.00    0    3.94
 *.akamai.net              4      0.00    12428    0.00    1    4.43
 other: 2 requested urlhosts        2      0.00     2021     0.00    1    0.71
192.168.2.101              63    0.00   295315   0.00    1     4.65
 cord.de                   17    0.00   115787   0.00    0   20.86
 *.doubleclick.net                  13    0.00    26163   0.00    1    2.07
 *.google.com              10    0.00    30646   0.00    1    3.71
 *.squid-cache.org         8      0.00    51758   0.00    1    6.53
 <error>                                 4     0.00     4290     0.00    0   10474
 other: 6 requested urlhosts        11    0.00    66671   0.00    5    2.28
--------------------------------- --------- ------ -------- ------ ---- -------
Sum                                    135    0.00   618651   0.00    1    6.51
```

- **Webalizer**

This tool is often installed by default on some Linux distributions. It is also GPL'ed and can be downloaded from http://www.mrunix.net/webalizer/. By editing the **/etc/webalizer.conf** file one can choose between apache access logs, ftp transfer logs or squid logs.

Example graphics generated with **webaliser**.



Reading

From the SuSE Linux Enterprise Server (Installation and Administration Document) read Pg 781 – 800 on Squid Proxy and SquidGuard Implementation on SUSE Linux.

You can read more about the Microsoft ISA Server with inbuilt proxy server here: http://www.microsoft.com/forefront/edgesecurity/isaserver/en/us/default.aspx. See this paper for the comparisons http://www.webperformanceinc.com/library/files/proxy_server_performance.pdf

Module summary

Summary

In this module you learned how to set up a Linux Webserver and Proxy server. These are common functions that you as the Linux System Administrator will be required to perform as from time to time.

Assignment

Assignment

Study the following two (2) articles and follow the instructions in them to set up a Linux based proxy server

- http://www.freeos.com/articles/2516/
- http://learnlinux.tsf.org.za/courses/build/electives/ch03.html

Module 18

Email Gateway

Introduction

Outcomes

Upon completion of this module you will be able to:

- Install and configure MTA's
- Implementing Spam and Content Filtering
- Configuring Virus Filtering with MTA's
- Operate and Perform Basic Configuration of Sendmail MTA's
- Deploying secure Webmail services

Terminology

MTA:	Mail Transport Agent. This is the mail component with the responsibility of deciding if mail handed to it is for a local account or not. It passes local mail to an MDA or stores it directly in mailstore itself. Remote mail is passed to another MTA.
MUA:	Mail User Agent. The client mail component which retrieves mail from mailstore and presents it to the user. It allows the user to create new mail and to send it to a MTA for onward transmission. Often the MUA will be associated with a graphical interface.

Mail Transfer Agent (MTA)

A mail transfer agent (MTA) (also called a mail transport agent, message transfer agent, or smtpd (short for SMTP daemon), is a computer program or software agent that transfers electronic mail messages from one computer to another. The term mail server is also used to mean a computer acting as an MTA that is running the appropriate software. An MTA receives mail from another MTA (relaying) or from a mail user agent (MUA). The MTA works behind the scenes, while the user usually interacts with the MUA.

For a listing of mail server see http://en.wikipedia.org/wiki/List_of_mail_servers . In this documentation we shall focus on the Sendmail Mail Server

Sendmail Configuration

Make sure that you install the necessary packages from the package manager or from the command line. Ensure also that your Domain Name Server is properly configured. We go into sendmail's main configuration directory **/etc/mail**. Here we need to do the following:

By default sendmail is configured to listen for connections ONLY for the 127.0.0.1 interface. In order to make sendmail listen to all interfaces we need to comment out the following line in **/etc/mail/sendmail.mc** using 'dnl' which stands for "do next line":

```
dnl  DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

Once this is done run:

```
m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf
```

Notice: Make sure /etc/sendmail.cf isn't also there, if it is, delete it. Restart sendmail and try the following:

```
telnet test1.seafront.bar 25
```

Warning: If you get a connection then sendmail is responding. This doesn't mean that sendmail will deliver mail (relay) for you!

To configure sendmail to relay for you you need to add the IP for your machine to the **/etc/mail/access** file:

```
192.168.246.12     RELAY
```

Finally, we also need to tell sendmail to accept mail for @seafront.bar addresses. For this, add the domain name to **/etc/mail/local-host-names**:

```
seafront.bar
```

Restart sendmail and send a mail to an existing user. If you have a user tux on the machine then check the output of the following:

```
mail -v -s "test seafront domain" tux@seafront.bar < /etc/passwd
```

We want the server seafront.bar to accept mail for the city.bar domain. For this we follow the following steps.

The DNS entries

We need to add an MX record for the city.bar domain. Here is the whole block for clarity:

```
seafront.bar.           IN      MX 10   test1.seafront.bar.

city.bar.               IN      MX 10   test1.seafront.bar.

test1.seafront.bar.     IN      A               192.168.246.12
```

Reload the zone file:

```
rndc reload
```

Sendmail Settings

1. We need to make sendmail accept mail for users at @city.bar. For this we add the next line to the **local-host-names** file:

```
city.bar
```

If mail is sent to *tux@city.bar* and *tux* is a valid user on test1.seafront.bar then mail will be delivered to the local user *tux*.

To avoid this we can use the **/etc/mail/virtusertable** database.

2. If you want to forward mail onto another account here are example entries for the **virtusertable** database:

```
tux@city.bar      mr.tux@otherdomain.org
@city.bar         administrator
list@city.bar     local-list
```

Here mail for user tux is diverted to mr.tux@otherdomain.org, the user administrator is the catchall account, lists are redirected to local lists (this needs to point to a valid list defined in the aliases

Using Procmail to receive mail

In depth information can be found in the **procmail, procmailrc** and **procmailex** manpages. Here are a few examples taken from **procmailex(5).** Sort all mail coming from the lpi-dev mailing list into the mail folder LPI:

```
:0:
* ^TO_lpi-dev
LPI
```

Forward mails between two accounts *main.address* and *the-other.address.* This rule is for the procmailrc on the main address account. Notice the X-Loop header used to prevent loops:

```
:0 c
        * !^X-Loop: yourname@main.address
        | formail -A "X-Loop: yourname@main.address" | \
          $SENDMAIL -oi yourname@the-other.address
```

The **c** option tells procmail to keep a local copy.

Reflection

One of the other popular Email MTA is the Microsoft Exchange Server, See http://www.microsoft.com/exchange/2007/evaluation/editions.mspx for a description of key features within the system. Note why you would implement each of the various versions.

Submit your comments and observations on your Wiki Page inside the http://www.colwiki.org/Reflections_and_Assignments webpage

Module summary

Summary

In this module you learned how to set-up your SMTP Server.It is important that you are also able to set-up your mail user agent and implement pop/imap services within your server. This can be done by using an POP/IMAP Server e.g imapd and dovecot.

Configuration of secure email access is one of the areas that Linux is quite strong and is an area where you shall be required to implement and manage time and time again.

Assignment

Assignment

Follow the instructions on the following two(2) websites and set-up your SMTP Server using postfix. You may also decide to use webmin for the configuration rather than setting up from the command line.

- http://rimuhosting.com/support/settingupemail.jsp?mta=postfix
- http://rimuhosting.com/knowledgebase/linux/mail/postfix%20notes

Install the dovecot package from the installation media and test with an appropriate Mail User Agent as to whether you can be able to send and receive email.

Module 19

Linux Security

Introduction

Outcomes

Upon completion of this module you will be able to:

- Perform Security Administration Tasks
- Set Up Host Security - Configure security environment files
- Set Up User-Level Security
- Minimization and Hardening of a Linux Server
- Setting up a Stateful IPTable Firewall
- Installing and Configuring of an Intrusion Detection System: Snort and Port Sentry
- Security Tools: SSH, LSOF, NETSTAT,TCPDUMP and NMAP

Terminology

Firewall:	A method of protecting the files and programs on one network from users on another network. A firewall blocks unwanted access to a protected network, while giving the protected network access to networks outside of the firewall.
BIOS:	The Basic Input/Output System (BIOS), also known as the System BIOS, is a de facto standard defining a firmware interface. The BIOS is boot firmware, designed to be the first code run by a PC when powered on. The initial function of the BIOS is to identify, test, and initialize system devices such as the video display card, hard disk, and floppy disk and other hardware. This is to prepare the machine into a known state, so that software stored on compatible media can be loaded, executed, and given control of the PC. This process is known as booting, or booting up, which is short for bootstrapping.

Host Security

System security is always a trade-off between convenience and features on the one hand and protectiveness and removing unnecessary risks on the other. As the cliché goes, security is inversely proportional to convenience: the easier a system is to use, the less secure it's likely to be. In contrast to many discussions in this column, this month we turn our attention to the "secure if inconvenient" end of the spectrum. Linux security will require the securing of both the host and the network (perimeter). Security the host can be performed through hardening. You may take the follwong steps to harden your linux box:

1. BIOS and Boot Loader Security: Password protection for the BIOS and the boot loader can prevent unauthorized users who have physical access to your systems from booting from removable media or attaining root through single user mode. But the security measures one should take to protect against such attacks depends both on the sensitivity of the information the workstation holds and the location of the machine. If an intruder has access to the BIOS, they can set it to boot off of a diskette or CD-ROM. This makes it possible for them to enter rescue mode or single user mode, which in turn allows them to seed nefarious programs on the system or copy sensitive data.
2. Administrative Controls : Do Not Use Only Words or Numbers — You should never use only numbers or words in a password.
    - Do Not Use Recognizable Words — Words such as proper names, dictionary words, or even terms from television shows or novels should be avoided, even if they are bookended with numbers.
    - Do Not Use Words in Foreign Languages — Password cracking programs often check against word lists that encompass dictionaries of many languages. Relying on foreign languages for secure passwords is of little use.
    - Do Not Use Hacker Terminology — If you think you are elite because you use hacker terminology — also called l337 (LEET) speak — in your password, think again. Many word lists include LEET speak.
    - Do Not Use Personal Information — Steer clear of personal information. If the attacker knows who you are, they will have an easier time figuring out your password if it includes your personal information.
    - Do Not Invert Recognizable Words — Good password checkers always reverse common words, so inverting a bad password does not make it any more secure.

Password Protecting GRUB

You can configure GRUB to address access of the prompt issues. To do this, first decide on a password, then open a shell prompt, log in as root, and type:

```
/sbin/grub-md5-crypt
```

When prompted, type the GRUB password and press [Enter]. This will return an MD5 hash of the password.

Next, edit the GRUB configuration file /boot/grub/grub.conf. Open the file and below the timeout line in the main section of the document, add the following line:

```
password --md5 <password-hash>
```

Replace <password-hash> with the value returned by /sbin/grub-md5-crypt[2]. The next time you boot the system, the GRUB menu will not let you access the editor or command interface without first pressing [p] followed by the GRUB password.

Unfortunately, this solution does not prevent an attacker from booting into a non-secure operating system in a dual-boot environment. For this you need to edit a different part of the /boot/grub/grub.conf file. Look for the title line of the non-secure operating system and add a line that says lock directly beneath it.

For a DOS system, the stanza should begin similar to the following:

```
title DOS
lock
```

You must have a password line in the main section of the /boot/grub/grub.conf file for this to work properly. Otherwise an attacker will be able to access the GRUB editor interface and remove the lock line. If you wish to have a different password for a particular kernel or operating system, add a lock line to the stanza followed by a password line. Each stanza you protect with a unique password should begin with lines similar to the following example:

```
title DOS
lock
password --md5 <password-hash>
```

Password aging is another technique used by system administrators to defend against bad passwords within an organization. Password aging means that after a set amount of time (usually 90 days) the user is prompted to create a new password. The theory behind this is that if a user is forced to change his password periodically, a cracked password is only useful to an intruder for a limited amount of time. The downside to password aging, however, is that users are more likely to write their passwords down.

The -M option of the chage command specifies the maximum number of days the password is valid. So, for instance, if you want a user's password to expire in 90 days, type the following command:

```
chage -M 90 <username>
```

In the above command, replace <username> with the name of the user. If you do not want the password to expire, it is traditional to use a value of 99999 after the -M option (this equates to a little over 273 years).

Disable root login over SSH. SSH brute force password attacks are arguably the biggest threat to Linux systems. There are tons of SSH brute force attacks running these days and all of your public facing servers should have the SSH port firewalled. If they absolutely must be accessible to the outside, consider setting up the firewall to restrict that outside access to the certain addresses you need, or provide access over a VPN.

Disable VNC administration. run VNC over an SSH tunnel, or use NX instead. VNC is unencrypted, so any passwords you type are sent over the wire in clear text.

Perimeter Security

Perimeter security is a primary consideration in any decision to host a website as the threats are becoming more widespread and persistent every day. One means of providing additional protection is to use a firewall.

There are two types of firewalls in Linux:

1. Iptables Firewall
2. Ip chains firewall.

These two firewalls shall be discussed simultaneously here.

Originally, the most popular firewall/NAT package running on Linux was ipchains, but it had a number of shortcomings. To rectify this, the Netfilter organization decided to create a new product called iptables, giving it such improvements as:

- Better integration with the Linux kernel with the capability of loading iptables-specific kernel modules designed for improved speed and reliability.
- Stateful packet inspection. This means that the firewall keeps track of each connection passing through it and in certain cases will view the contents of data flows in an attempt to anticipate the next action of certain protocols. This is an important feature in the support of active FTP and DNS, as well as many other network services.
- Filtering packets based on a MAC address and the values of the flags in the TCP header. This is helpful in preventing attacks using malformed packets and in restricting access from locally attached servers to other networks in spite of their IP addresses.
- System logging that provides the option of adjusting the level of detail of the reporting.
- Better network address translation.
- Support for transparent integration with such Web proxy programs as Squid.
- A rate limiting feature that helps iptables block some types of denial of service (DoS) attacks.

Considered a faster and more secure alternative to ipchains, iptables has become the default firewall package installed under many Linux distributions. Iptables can be installed with the relevant package manager available in your distribution

You can start, stop, and restart iptables after booting by using the commands:

```
[root@linux]# service iptables start
[root@linux]# service iptables stop
[root@linux]# service iptables restart
```

To get iptables configured to start at boot, use the chkconfig command:.

```
[root@linux]# chkconfig iptables on
```

You can determine whether iptables is running or not via the service iptables status command. For example

```
[root@linux]# service iptables status
Firewall is stopped.
[root@linux]#
```

All packets inspected by iptables pass through a sequence of built-in tables (queues) for processing. Each of these queues is dedicated to a particular type of packet activity and is controlled by an associated packet transformation/filtering chain.

There are three tables in total. The first is the mangle table which is responsible for the alteration of quality of service bits in the TCP header.

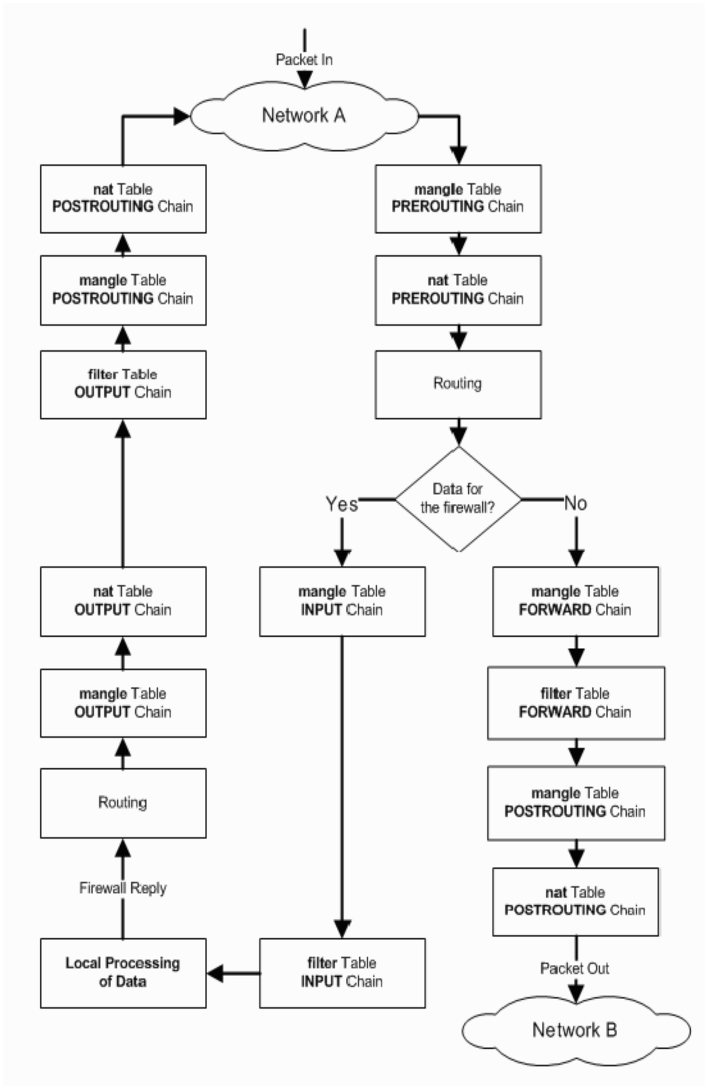
The second table is the filter queue which is responsible for packet filtering. It has three built-in chains in which you can place your firewall policy rules. These are the:

- Forward chain: Filters packets to servers protected by the firewall.
- Input chain: Filters packets destined for the firewall.
- Output chain: Filters packets originating from the firewall.

The third table is the nat queue which is responsible for network address translation. It has two built-in chains; these are:

- Pre-routing chain: NATs packets when the destination address of the packet needs to be changed.
- Post-routing chain: NATs packets when the source address of the packet needs to be changed

The chart below summarises this.


Packet In
Network A
mangle Table PREROUTING Chain
nat Table PREROUTING Chain
Routing
Data for the firewall?
Yes
No
mangle Table INPUT Chain
filter Table INPUT Chain
Local Processing of Data
Firewall Reply
Routing
mangle Table OUTPUT Chain
nat Table OUTPUT Chain
filter Table OUTPUT Chain
mangle Table POSTROUTING Chain
nat Table POSTROUTING Chain
mangle Table FORWARD Chain
filter Table FORWARD Chain
mangle Table POSTROUTING Chain
nat Table POSTROUTING Chain
Packet Out
Network B

Each firewall rule inspects each IP packet and then tries to identify it as the target of some sort of operation. Once a target is identified, the packet needs to jump over to it for further processing. Table below lists the built-in targets that iptables uses.

Target	Desciption	Most Common Options

ACCEPT	• iptables stops further processing. • The packet is handed over to the end application or the operating system for processing	N/A
DROP	• iptables stops further processing. • The packet is blocked	N/A
LOG	• The packet information is sent to the syslog daemon for logging • iptables continues processing with the next rule in the table • As you can't log and drop at the same time, it is common to have two similar rules in sequence. The first will log the packet, the second will drop it.	--log-prefix "string" Tells iptables to prefix all log messages with a user defined string. Frequently used to tell why the logged packet was dropped
REJECT	• Works like the DROP target, but will also return an error message to the host sending the packet that the packet was blocked	--reject-with qualifier The qualifier tells what type of reject message is returned. Qualifiers include: icmp-port-unreachable (default) icmp-net-unreachable icmp-host-unreachable icmp-proto-unreachable icmp-net-prohibited icmp-host-prohibited tcp-reset echo-reply
DNAT	• Used to do **destination network address translation.** ie. rewriting the destination IP address of the packet	--to-destination ipaddress Tells iptables what the destination IP address should be
SNAT	• Used to do **source network address**	--to-source <address>[-<address>][:<port>-

	translation rewriting the source IP address of the packet • The source IP address is user defined	<port>] Specifies the source IP address and ports to be used by SNAT.
MASQUERADE	• Used to do Source Network Address Translation. • By default the source IP address is the same as that used by the firewall's interface	[--to-ports <port>[-<port>]] Specifies the range of source ports to which the original source port can be mapped.

Each line of an iptables script not only has a jump, but they also have a number of command line options that are used to append rules to chains that match your defined packet characteristics, such the source IP address and TCP port. There are also options that can be used to just clear a chain so you can start all over again. The tables below list the most common options.

iptables command Switch	**Desciption**
-t <-table->	If you don't specify a table, then the filter table is assumed. As discussed before, the possible built-in tables include: filter, nat, mangle
-j <target>	Jump to the specified target chain when the packet matches the current rule.
-A	Append rule to end of a chain
-F	Flush. Deletes all the rules in the selected table
-p <protocol-type>	Match protocol. Types include, icmp, tcp, udp, and all
-s <ip-address>	Match source IP address
-d <ip-address>	Match destination IP address
-i <interface-name>	Match "input" interface on which the packet enters.

-o <interface-name>	Match "output" interface on which the packet exits

In this command switches example

```
iptables -A INPUT -s 0/0 -i eth0 -d 192.168.1.1  -p TCP -j ACCEPT
```

iptables is being configured to allow the firewall to accept TCP packets coming in on interface eth0 from any IP address destined for the firewall's IP address of 192.168.1.1. The 0/0 representation of an IP address means any.

Switch	Desciption
-p tcp --sport <port>	TCP source port. Can be a single value or a range in the format: *start-port-number:end-port-number*
-p tcp --dport <port>	TCP destination port. Can be a single value or a range in the format: *starting-port:ending-port*
-p tcp --syn	Used to identify a new TCP connection request. ! --syn means, not a new connection request
-p udp --sport <port>	UDP source port. Can be a single value or a range in the format: *starting-port:ending-port*
-p udp --dport <port>	UDP destination port. Can be a single value or a range in the format: *starting-port:ending-port*

In this example:

```
iptables -A FORWARD -s 0/0 -i eth0 -d 192.168.1.58 -o eth1 -p TCP \
         --sport 1024:65535 --dport 80 -j ACCEPT
```

iptables is being configured to allow the firewall to accept TCP packets for routing when they enter on interface eth0 from any IP address and are destined for an IP address of 192.168.1.58 that is reachable via interface eth1. The source port is in the range 1024 to 65535 and the destination port is port 80 (www/http).

You can configure iptables to have user-defined chains. This feature is frequently used to help streamline the processing of packets. For example, instead of using a single, built-in chain for all protocols, you can use the chain to determine the protocol type for the packet and then hand off the actual final processing to a user-defined, protocol-specific chain in the filter table. In other words, you can replace a long chain with a stubby main chain pointing to multiple stubby chains, thereby shortening the total length of all chains the packet has to pass through. For example

```
iptables -A INPUT -i eth0  -d 206.229.110.2 -j fast-input-
queue
```

```
iptables -A OUTPUT -o eth0 -s 206.229.110.2 -j fast-output-
queue

iptables -A fast-input-queue  -p icmp -j icmp-queue-in
iptables -A fast-output-queue -p icmp -j icmp-queue-out

iptables -A icmp-queue-out -p icmp --icmp-type echo-request
\
         -m state --state NEW -j ACCEPT

iptables -A icmp-queue-in -p icmp --icmp-type echo-reply -j
ACCEPT
```

Here six queues help assist in improving processing speed

INPUT	The regular built-in INPUT chain in iptables
OUTPUT	The regular built-in OUTPUT chain in iptables
fast-input-queue	Input chain dedicated to identifying specific protocols and shunting the packets to protocol specific chains.
fast-output-queue	Output chain dedicated to identifying specific protocols and shunting the packets to protocol specific chains.
icmp-queue-out	Output queue dedicated to ICMP
icmp-queue-in	Input queue dedicated to ICMP

The service iptables save command permanently saves the iptables configuration in the /etc/sysconfig/iptables file. When the system reboots, the iptables-restore program reads the configuration and makes it the active configuration.

The initialization of built-in chains is automatic and the string "iptables" is omitted from the rule statements.

Here is a sample /etc/sysconfig/iptables configuration that allows ICMP, IPSec (ESP and AH packets), already established connections, and inbound SSH.

```
[root@linux]# cat /etc/sysconfig/iptables
# Generated by iptables-save v1.2.9 on Mon Jun 8 11:00:07 2009
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [144:12748]
:RH-Firewall-1-INPUT - [0:0]
```

```
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp -m icmp --icmp-type 255 -j ACCEPT
-A RH-Firewall-1-INPUT -p esp -j ACCEPT
-A RH-Firewall-1-INPUT -p ah -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
# Completed on Mon Nov 8 11:00:07 2004
[root@linux]#
```

Linux Security Tools

SSH

sshd_con fig overview

Port 22	Specify which port to listen on. Multiple “Port” options can be used
Protocol 2,1	Specify version 1 or version 2. Can be a comma separated list. If both are supplied, they are tried in the order presented.
DenyUsers [USER]@HOST	Deny users from a specific host. Wild cards such as * can be used
PermitRootLogin yes/no	Allow or disallow root access
X11Forwarding yes/no	Instructs the remote end to route X11 traffic back through the ssh tunnel to the user's X session. Unless disabled, the xauth settings will be transferred in order to properly authenticate remote X applications

LSOF

lsof - show open files used by processes

Traditionally used to list PIDs of processes running on a given directory:

```
lsof +D DIRECTORY
```

lsof will output the following information:

NAME:	name of the process
PID:	process ID
USER:	name of the user to whom the process belongs
FD:	File desciptor (e.g u = read write, r = read, w = write)
TYPE:	The file type (e.g REG = regular file)
DEVICE:	Major/Minor number (e.g 3,16 =/dev/hda16)
SIZE:	Size or offset of the file
NODE:	Inode of the file
NAME:	The name of the file

Lsof can also be used to display network sockets. For example the following line will list all internet connections:

```
lsof -i
```

You can also list connections to a single host:

```
lsof -i @HOST
```

For example if a host TOFFY is connected to your localhost on port 1234, the following would display information about the connection:

```
lsof -i @TOFFY:1234
```

NETSTAT

netstat - Print network connections, routing tables ...

Main options are:

-r	display routing tables
-l	only listening services
-C	display route cache
--inet	restrict to network sockets

NMAP

nmap - Network exploration tool and security scanner

The scanner makes use of the fact that a closed port should (according to RFC 793) send back an RST. In the case if a SYN scan, connections that are half opened are immediately close by nmap by sending an RST itself.

Scan Types:

SYN or Half-open: -sS
Nmap will send a synchronisation packet SYN asking for a connection. If the remote host send a RST/ACK it is assumed that the port is closed. If the remote host sends a SYN/ACK this indicates that the port is listening.

UDP: -sU
UDP is connectionless. So there is no need for a 3 way handshake as with TCP. If a port is closed the server will send back a ICMP PORT UNREACHABLE. One then deduces that all the other ports are open (not reliable in the case were ICMP messages are blocked).

TCP NULL: -sN
TCP packet with no flags set. Closed port will send a RST when receiving this packets (except with MS Windows).

TCP Xmas: -sX
TCP packet with the FIN+URG+PUSH flags set. The remote host should send back a RST for all closed ports when receiving a Xmas packet.

Reading

From the SuSE Linux Enterprise Server (Installation and Administration Document) read Pg 819 – 843 on Firewall Implementation on SUSE Linux

Reflection

Microsoft Windows Firewall is implemented through the ISA. See this documentation for ISA (http://www.microsoft.com/forefront/edgesecurity/isaserver/en/us/default.aspx) and see if there are significant differences. Note them and post them within your private space in this books webite.

Module summary

In this module you learned above various Linux based firewalls and other security measures that you can take on a Linux based system. The configurations for the Linux system may be tedious and therefore I would ask

Summary

you to use the webmin Linux Firewall interface to configure your firewall.

Assignment

Assignment

Use the webmin interface installed in a previous exercise to configure your Linux firewall

Module 20

Case Study – Installing Moodle Learning Management System

Introduction

In this module you will use the skills learned in this course to install the Moodle Learning Management System on both a Linux and Windows based Apache + MySQL + PHP platform. More specifically, you will perform the following:

- Install Moodle Pre-requisites
- Install the moodle Learning Management System

Outcomes

Terminology

LMS:	The Learning Management System is Software that automates the administration of training. The LMS registers users, tracks courses in a catalog, records data from learners; and provides reports to management. It is software for delivering, tracking and managing training/education.
Moodle:	Moodle is a free and open source e-learning software platform, also known as a Course Management System, Learning Management System, or Virtual Learning Environment. It has a significant user base with 49,256 registered sites with 28,177,443 users in 2,571,855 courses (as of February, 2009). Moodle is designed to help educators create online courses with opportunities for rich interaction. Its open source license and modular design means that people can develop additional functionality.

Installing Moodle in Linux

Moodle requires the following pre-requisites:

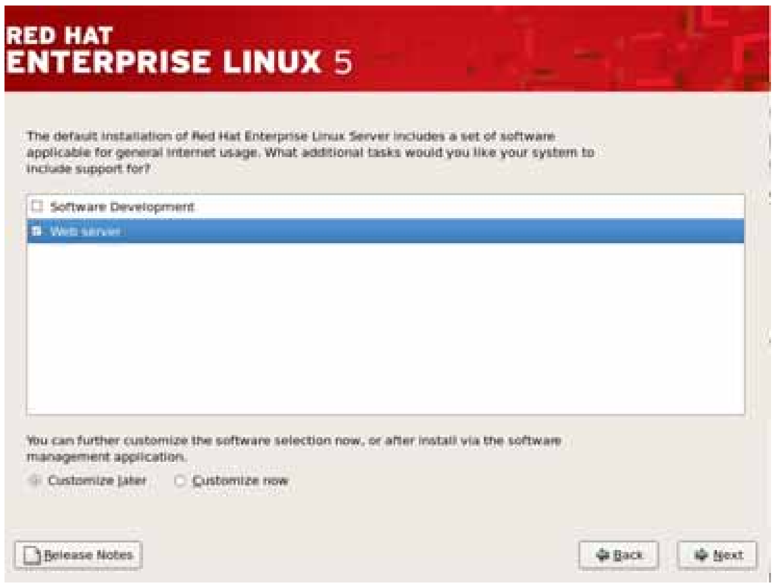
1. Apache Web Server: Moodle is a Web Based system requiring the user to access the system via a web browser. Though Moodle can

work on any webserver supporting PHP (including IIS), many installations use the Apache Webserver. Apache is short for Apache HTTP Server Project, a robust, commercial-grade, featureful, and freely-available open source HTTP Web Server software produced by the Apache Software Foundation. It is the most commonly used web server on the internet, and is available on many platforms, including Windows, Unix/Linux, and Mac OS X.

2. MySQL: A Learning Management System (LMSA mail transfer agent (MTA) (also called a mail transport agent, message transfer agent, or smtpd (short for SMTP daemon), is a computer program or software agent that transfers electronic mail messages from one computer to another. The term mail server is also used to mean a computer acting as an MTA that is running the appropriate software. An MTA receives mail from another MTA (relaying) or from a mail user agent (MUA). The MTA works behind the scenes, while the user usually interacts with the MUA.MySQL is a popular open source SQL (Structured Query Language) database implementation, available for many platforms, including Windows, Unix/Linux and Mac OS X.

3. PHP: PHP is a recursive acronym for PHP: Hypertext Preprocessor. It is a popular server-side scripting language designed specifically for integration with HTML, and is used (often in conjunction with MySQL) in Content Management Systems and other web applications. It is available on many platforms, including Windows, Unix/Linux and Mac OS X, and is open source software.

We will install these pre-requisites on a Red-Hat Linux Box. During installation of RHEL 5 make sure that you install the Webserver option as shown in the screen below:



The default installation of Red Hat Enterprise Linux Server includes a set of software applicable for general internet usage. What additional tasks would you like your system to include support for?

- [ ] Software Development
- [x] Web server

You can further customize the software selection now, or after install via the software management application.

[✓ Customize later] Customize now

APACHE INSTALLATION

It is advisable that you install nmap package useful to determine the running services on your linux box. To install nmap navigate to your instlallation media as follows:

```
$cd /media/CDROM/Server

$rpm –i nmap-4.11-1.1.i386.rpm
```

This will install the nmap package of which you can run

```
$nmap localhost
```

Or

```
Nmap <ip_address_of_your_server>
```

And determine the servers that you are running. You should notice that your port 80 (HTTP Webserver is not running). You can start the apache webserver by running:

```
$/etc/init.d/httpd start
```

This will start your apache server. You can perform nmap on the ip address of your host to confirm.

MYSQL INSTALLATION

To install MySQL Server, you will need to install the perl-DBI dependency. This can be done as follows:

```
$rpm –i  perl-DBI-1.52-1.fc6.i386.rpm mysql-5.0.22-2.1.i386.rpm

$rpm –i perl-DBD-MySQL-3.0007-1.fc6.i386.rpm mysql-server-5.0.22-2.1.i386.rpm
```

You can now start the MySQL Service by running:

```
$/etc/init.d/mysqld start
```

By default the MySQL database does not come with a superuser password ('root user'). This is a security risk that you need to seal. To do this run:

```
$mysqladmin –u root password 'root2009';
```

Replace root2009 with your required root password.

Then restart your mysql database by running:

```
$/etc/init.d/mysqld restart
```

Connect to MySQL by running the following:

```
$mysql –u root –p
```

Enter the MySQL Password you just created, then press enter, you will see the MySQL Prompt. Create the moodle database by running:

```
$create database moodle;
```

MOODLE INSTALLATION

Get the moodle packages from http://download.moodle.org/

```
$wget http://download.moodle.org/download.php/stable19/moodle-1.9.5.tgz
```

Then copy the file to the webserver root directory:

```
$cp moodle-1.9.5.tgz /var/www/html/
```

Uncompress as follows

```
$tar –zxvf moodle-1.9.5.tgz
```

A directory named moodle will be created in your current working directory which should be /var/www/html/

Assign the file the appropriate rights by:

```
$chown –R apache:apache moodle
```

Create moodledata directory required by moodledata

```
$mkdir –p /var/www/html/moodledata

$chown –R apache:apache  moodledata
```

Open up your browser and access the moodle installer from the browser as follows:

http://<ipaddress_of_you_computer>/moodle/

Follow through the installation instructions in the installer. Make sure you specify the mysql database as the database you have created while setting up MySQL.

See http://docs.moodle.org/en/Installing_Moodle for more information and further installation procedures in other Linux/Operating System.

Reading

Installing Moodle in Windows

Installation of the pre-requisites in Windows is made easy through the use of pre-compiled and pre-configured packages that automatically install a pre-configured Apache+MySQL+PHP.There are many packages of this nature available, but the author of this course guide recommends XAMPP from “apachefriends”.

1. For instructions to install Apache, MySQL and PHP on Windows follow the Instructions on http://docs.moodle.org/en/Windows_installation_using_XAMPP. This will install the pre-requisites required for moodle installation.

2. To install moodle on Windows follow the Instructions on the Moodle Site: http://docs.moodle.org/en/Complete_install_packages_for_Windows