Enterprise Linux Systems Administration I with a SUSE Linux

(SUSE Linux Enterprise Server and Desktop, OpenSUSE)

bias

Table of Contents

Talk to us!....6

Introduction....7

IRC Lab....9

BASH keyboard shortcuts....10

Basic Regular Expressions....13

Managing Users and Groups....18

Managing Users and Groups Lab....20

A Proper Way To Edit....22

Elevating privilege....26

SUDO Lab....30

ISCSI....33

Managing Storage....34

Disk Management (Physical Partitions)....34

Disk management tools....36

fdisk....36
cfdisk....38
parted....38
mkfs....40
Using UUID's....42

Swap Space....43

Managing Storage Labs....46

Logical Volume Management....49

Physical Volumes....49

Volume Groups....50

Logical Volumes....52

Adding More Disks....55

Growing a Logical Volume....56

Reducing a Logical Volume....58

Removing a disk from the LVM framework....60

Logical Volume Management Labs....61

Cool filesystem tools....63

Normal POSIX Permissions....66

Regular (POSIX) Permissions....66

Normal Posix Permissions Labs....71

Special Permissions....72

Set User ID bit....72

Set Group ID bit....73

Sticky Bit....75

Summary....76

ACLs (Access Control Lists)....79

Viewing an ACL....79

Setting an ACL....79

ACL's and inheritance....81

The mask....84

ACL Labs....86

Filesystem Layouts....87

The basics....87

Filesystem Labs....90

Server specific configurations....91

HTTP and FTP servers:....91
Proxy servers:....91
Mail servers:....91
Database servers:....92

Managing Networking....93

NetworkManager....93

Traditional Networking....93

Bridging....95

Routing....96

Setting your system's hostname....97

Networking tools....97

Networking Labs....98

Process Management....100

Process Management Labs....102

Monitoring Resources....104

CPU....104
Memory....105
Disk....105
Network....106
Monitoring Resources Labs....107
Software Management on SUSE Linux....108
Software Management on SUSE Linux Labs....114
Service Management....115
Service Management Labs....117
autofs....118
Filtering traffic using the firewall....121
yast firewall for SUSE Linux....121
Secure Communication....122
SSH....122
Establishing an SSH session....123
Configuring SSH to use Public/Private key authentication....126
Configuring the SSH daemon....128
Basic security precautions and hardening SSHd....128
telnet....131
Secure Communications Labs....132
Virtualization....134
KVM....134

QEMU.....135
Paravirtualization vs Full Virtualization.....135
Creating a Virtual Machine.....135
Starting Virtual Machines.....137

Virtualization Labs.....139

Logging.....140

Facilities.....140

Priorities.....142

Logging Labs.....146

Changing Kernel Settings.....147

/proc/sys and /sys.....147

/etc/sysctl.conf.....147

/etc/grub.conf.....147

/etc/modprobe.d/something.conf.....148

/etc/rc.local.....148

kernel-doc.....148

Changing Kernel Settings Labs.....149

Talk to us!

Karl Clinger	karl.clinger@enterpriselinux.pro
Ricardo da Costa	rgdacosta@enterpriselinux.pro
Curriculum Support	curriculum-support@enterpriselinux.pro
IRC Chat	chat.enterpriselinux.pro
Facebook	http://www.facebook.com/enterpriselinux
Google+	http://goo.gl/4YtkMe
Twitter	http://www.twitter.com/EntLinuxPros
Blog	http://blog.enterpriselinux.pro
Website	http://www.enterpriselinux.pro

Introduction

Welcome to class!

Our objective is to guide and facilitate your Enterprise Linux learning. We have a wonderful global support network to assist you in this.

We have tips and important notes which are separated from the curriculum text in blocks which appear as follows:

!!! TIP !!!

And

!!! NOTE !!!

Any command which should be typed as a **normal user** would be featured in a block and the prompt will being with a **$**, for example:

```
$ command
```

Any command which should be typed as the **root user** would also be featured in a block and the prompt with begin with a **#**, for example:

```
# command
```

Support options available:

Karl Clinger	karl.clinger@enterpriselinux.pro
Ricardo da Costa	rgdacosta@enterpriselinux.pro
Curriculum Support	curriculum-support@enterpriselinux.pro
IRC Chat	chat.enterpriselinux.pro

Should you wish to connect to our chat server, feel free to make use of the following clients:

Linux	XChat
Windows	mIRC
Macintosh	Colloquy

We'll begin right away with some basics which you should be aware of. Have a look right now at the labs document and complete pages 1 - 10.

Should you need some assistance use our IRC chat server and join the #linux channel to ask for help.

!!! TIP 1 !!!

To connect to an IRC server use the command /server chat.enterpriselinux.pro

!!! TIP 2 !!!

To join a channel use the command /join #linux

!!! TIP 3 !!!

To send a private message use the command /msg nickname message

IRC Lab

There’s only one way to move and that’s forward!

Have a look at the following scenarios during our initial phase and complete them. If you need hints or any form of assistance join us on chat.

Use a chat client and connect to:

Server Name	chat.enterpriselinux.pro
Port	6667
Channel	#linux

BASH keyboard shortcuts

These have all been tested with BASH 4.1 in an ncurses based environment. If you are using the terminal application then you may get unexpected results when using the **ALT** key.

> **!!! NOTE !!!**
>
> The key combinations listed below are to be hit simultaneously.

To see all the key bindings use the command **bind -P**

Tab	Autocompletes from the cursor position
Ctrl c	Sends the signal SIGINT to the current task, which aborts and closes it.
Ctrl d	Sends an EOF marker, which (unless disabled by an option) closes the current shell. It's the same as typing **exit** or **logout**

Ctrl h	Same as backspace
Ctrl i	Same as Tab
Ctrl j	Same as Enter
Ctrl l	Same as the **clear** command
Ctrl n	Scroll through the commands from the beginning
Ctrl p	Scroll through commands from the end
Ctrl r	Search command history
Ctrl u	Clears the line content before the cursor position
Ctrl w	Deletes the word before the cursor position
Ctrl x Ctrl x	Switches to the beginning of a line or to the end of a line
Ctrl x Ctrl e	Edits the current line using the editor defined by the variable $EDITOR

Ctrl x Ctrl v	Shows the version of BASH being used
Ctrl z	Sends the signal **STOP** to the current task which suspends it in the background. To take it out of the background and back into the foreground use either **fg** or send signal **CONT** to the process
Alt b or Esc b	Moves the cursor back one word
Alt d or Esc d	Cuts the word after the cursor
Alt f or Esc f	Moves the cursor one word forward
Alt l or Esc l	Lowers the case of every case of every character in a word and moves to the next word
Alt u or Esc u	Capitalizes the case of every case of every character in a word and moves to the next word
Alt . or Esc .	Insert the last argument to the previous command (the last word of the previous history entry)

Basic Regular Expressions

Many new Linux administrators find regular expressions rather intimidating but here we will start off slowly and work our way up to more complex examples in ELSAII.

Regular expressions are patterns which match text. That's it! They aren't programming languages or some dialect of Klingon. You simply tell your computer to show you where a pattern is found. For simplicity's sake, we sometimes call them **regex** instead of using its full name.

In this class we will focus on Basic Regular Expressions which we abbreviate to BRE so let's have a look at BRE examples:

Regex:
Simple text

What it means:
We literally look for the pattern **mentioned** everywhere. No regard is given for what comes before or after, as long as the letters **in the pattern** are together with no spaces between them, we will have a match.

Regex example:
ric

What it matches to:
ricardo
e**ric**
richard
e**ric**a
ricepaper

Regex:
\

What it means:
This takes away the special meaning of the next character and is used to treat the next character literally.

Regex example:
US\$

In the case above, a $ in regex means something else so here we don't want its special meaning used, we're literally looked for a dollar symbol.

What it matches to:
US$1 = ZAR10

Regex:
[ ]

What it means:

This is an example of a set and we match to only 1 element at a time inside the set. The order of the elements in the set isn't important.

Regex example:
r[iou]b

What it matches to:

rib
asc**rib**e
t**rib**ute
rob
robotically
uncor**rob**orated
rub
shr**ub**bery
che**rub**

Regex:
[^]

What it means:
Do *not* match to any of the elements in the set. Matching is done one at a time and the order of the elements is not important.

Regex example:
r[^iu]b

What it matches to:
rob
robotically
uncor**rob**orated
rebel
c**rab**
admi**rab**le

Regex:

.

What it means:
Match to any single character

Regex example:
gr.y

What it matches to:
grey
gray

Regex:
*

What it means:
Match to **zero *or* more** of the previous element

Regex example:
co*l

What it matches to:
color
colorful
cool
cooooooooooooooool
clear
cluster

Regex:
^

What it means:
This is called an anchor and it will match your regex only to the beginning of a line.

Regex example:
^The

What it matches to:
Any line which begins with **The** would be matched in this case.

Regex:
$

What it means:
This is called an anchor too and will match your regex to the end of a line.

Regex example:
bash$

What it matches to:
Any line which ends with **bash** will be matched.

Regex:
\<

What it means:
Match to the beginning of a word.

Regex example:
\<ba
What it matches to:
Any word that begins with **ba** will be matched regardless of where it is in a line.

Regex:
\>

What it means:
Match to the end of a word.

Regex example:
sh\>

What it matches to:
Any word that ends in **sh** will be matched regardless of where it is in a line.

Managing Users and Groups

Adding Users/Groups

Adding users and groups is simple enough. **useradd user1** would add the user user1 to your system. For this purposes of this class, we will explore what happens when a user is added and discuss the configuration files involved.

User and Group Configuration Files

When the useradd command is run, an entry is appended (added at the end) to the /etc/passwd, /etc/shadow and /etc/group files. Here is the breakdown of information about the new user:

/etc/passwd- This is where local account information is held.
/etc/shadow- This is where account authentication and expiration information is held. It is a companion file to /etc/passwd.
/etc/group- This is where local group information is held.

If the new user will be given additional privileges, an additional file must be configured.
/etc/sudoers- Regular users are granted elevated privileges in this file.

/etc/passwd

A detailed explanation of many configuration files can be found in the man pages. Chapter 5 is specifically “File formats and conventions.” For example, to see the “File format” for /etc/passwd, run the command:

```
$ man 5 passwd
```

Here is the root line from the **/etc/passwd** file:

```
root:x:0:0:root:/root:/bin/bash
```

Here is the explanation from man 5 passwd:

```
account:password:UID:GID:GECOS:directory:shell
```

/etc/shadow

Similarly, information about sensitive information can be found in **/etc/shadow**:

```
root:$6$SlbfFmmi$(hashed password):15218:0:99999:7:::
```

The /etc/shadow file columns are divided by colons. Going by this, there are 9 columns shown in the output above. The output from the command **man 5 shadow** offers the following explanation:
1. login name
2. encrypted password
3. date of last password change
4. minimum password age
5. maximum password age
6. password warning period
7. password inactivity period
8. account expiration date
9. reserved field

/etc/group

In Linux, every user is automatically given its own group. The configuration file for groups is much simpler than the user configuration files. Here is what **man 5 group** shows us about the format of **/etc/group**:

1. group_name
2. password
3. GID
4. user_list

Very rarely is a password used for groups. In fact, there is a little known file called **/etc/gshadow** for holding this information. This reduces what you need to know about this file to the name of the group, the group id (which usually lines up with the corresponding user id) and the members of the group.

Managing Users and Groups Lab

Open a terminal and determine the current username by using the command **whoami**

Answer	

Determine the user ID of that user by using **id**

Answer	

To assume the identity of the user root, use the command **su -**, now determine the root user's ID

Answer	

Now use the **exit** command to close the session where you are working as root. What is the username of the user which you are using now?

Answer	

Add the following groups and users with the properties in the table below:

Username	**Full Name**	**UID**	**Additional Groups**	**Shell**	**Home Directory**	**Password**
fred	Fred Flintstone	2000	flintstones, wbs	/bin/bash	/home/fred	bedrock60
wilma	Wilma Flintstone	3000	flintstones	/sbin/nologin	/dev/null	bedrock80

barney	Barney Rubble	4000	rubbles, wbs	/bin/bash	/home/barney	bedrock20
betty	Betty Rubble	5000	rubbles, wheel	/bin/ksh	/home/betty	bedrock30

Groupname	**GID**
flintstones	7000
wbs	9000
rubbles	8000

Users are to change their passwords on first login and all passwords expire every 28 days with the users receiving a warning of this 5 days before the time. Should a password expire then the user can log in with the expired password for 1 additional day.

Use the man pages for these commands to get inspiration:

useradd
usermod
groupadd
passwd
chage

A Proper Way To Edit

If two administrators open the same file and edit it at the same time the last admin who saves his or her changes wins. That is to say that the first admin has his or her changes overwritten. To avoid this occurrence, there are a few utilities open certain files as a copy, or temporary file, so that only the changes get applied to the original file. In most cases, this will avoid conflicts. It is a good idea to follow “best practices” when it is possible to do so. Here are the ways to edit the previously mentioned files as a best practice.

vipw

This utility opens a copy of the /etc/passwd file for editing and applies the changes to the original file.

vipw -s

This is how to open the /etc/shadow file for editing. Prior to EL6, whenever the vipw command was used and a save was applied the user would automatically be asked if the user would also like to change the /etc/shadow file.

vigr

Similar to vipw, but vigr opens a copy of the /etc/group file for editing and applies the changes to the original.

/etc/skel

At a glance, the /etc/skel directory seems empty. However, upon further inspection you will find several very important hidden files in this directory. (Remeber, files that start with a period are ordinarily invisible to a regular ls command. The “-a” option also shows hidden files.) The files in this file include:

.bash_logout
.bash_profile
.bashrc

The purpose of /etc/skel is to hold the files that populate home directories of new users. When you add a user you will notice that the files in their home directory are identical to the ones in /etc/skel. Rather than creating new users and then editing all of these files in their home directories, change the files in /etc/skel and then create the users. You can potentially save yourself a lot time.

/etc/login.defs

Many of the default settings for new users are in this file. Every uncommented line includes a setting. Here are some of the things this file does:

1. Creates a local mail spool directory for each user. For example, creating the user ricardo would also create /var/spool/mail/ricardo. This is where mail is stored until the user accesses it through a mail client (Mail User Agent, or MUA).
2. Specifies default account expiration information (stored in /etc/shadow).
3. Sets the user id and group id ranges.
4. Sets whether a home directory should be created for the new user. (Default is “Yes.”)
5. Creates a umask value for the new user.
6. Gives userdel the ability to remove empty groups.
7. Selects which encryption type to use for user passwords.

User and Group Management Commands

Now we will explore some of the commands used to manage users and groups. Although all of what we will discuss in this section can be accomplished by knowing

what needs to be edited in the configuration files, commands make it much easier. Commands also allow for automation of tasks.

useradd

This is the command to add a user. By default, a group by the same name as the user is created. The only member of this group, by default, is the new user. Some of the options to this command are useful to be aware of. Sometimes you do not want a user to be able to interact through a shell. Perhaps a users home directory needs to be somewhere other than /home, or maybe the user will not have a home directory at all.

The useradd command makes an entry in the /etc/passwd file, as follows for the user karl:

```
karl:x:500:500::/home/karl:/bin/bash
```

usermod

Once an entry is in the /etc/passwd file, it can be modified using the usermod command. The fifth field is known as the GECOS field (or comment). To change this information without editing the /etc/passwd file I could run the command:

```
usermod -c me karl
```

This puts a comment “me” in the /etc/passwd file, like this:

```
karl:x:500:500:me:/home/karl:/bin/bash
```

Other notable options are:

-a to append the user to a secondary group (always done with the “G” option in order to retain primary group membership in the users corresponding group).

-d to change the home directory of the user. (Used with -m to move the contents of the old home directory to the new one.

-s sets the users login shell. (There is really no good reason to change it from bash, unless the user prefers a different shell from a previous environment.)

!!! Note !!! The options for usermod are almost identical to the options for useradd.

passwd

By default, users do not have a password set when they are create. So, an addition step is required to set a password. The command to do this is **passwd**.

A question that arises pretty often is about the spelling of this command. This is where knowing a bit of history helps. The **passwd** command originated in Unix. There was a limit of characters per filename. Since most commands had a ".c" extension on them (i.e. passwd.c), that only left 6 characters remaining. So, *password* was shortened to *passwd* and *unmount* was shortened to *umount*, etc.

The passwd command makes an entry in the /etc/shadow file based on the settings in the /etc/login.defs file. The passwd command affects the second column (highlighted in red).
Here is a line from /etc/shadow for the user karl:

karl:**$**6**$**OnioiQp5Wnqw1tOj**$**nU9HPoSdePFlGvQ<truncated>:15217:0:99999:7:::

Within this column, there are three sections, designated by dollar signs ($). The first section specified what type of encryption is being used. The second section is "salt"

that was generated by the system for obfuscation. The third section is the actual hashed password of the user.

User Limitations

Linux administration of users is all about maintaining a balance of providing enough capabilities for users to do what needs to be done while, at the same time, restricting access to certain resources. Here are some commands that help to secure systems while allowing users to do their jobs.

chage

Account expiration is a way of managing users and ensuring that password information does not become stagnant. The configuration file for account expiration is /etc/shadow. The default values are stored in /etc/login.defs.

Elevating privilege

su

su stands for ‘substitute user’ and is used to substitute one’s user ID for that of another’s.

If you are logged on as root then you can substitute your user ID for that of any user without knowing that user’s password. However, if you’re logged on as a normal user then you would need to know the password of the user whose identity you want to assume.

Similarly, typing in su without specifying a username assumes that you want to assume the identity of root and will prompt you for root’s password:

```
$ whoami
fred
```

```
$ su -
Password:

# whoami
root
```

To completely assume the identity of another user use the **-** or **-l** options when using the su command. This starts a login shell for the user whose identity you want to assume which means that all the configuration files that would have been processed if you initially logged on as that user, will be processed now.

Failing to include the hyphen causes certain variables, aliases and functions to not be available to you even though your identity was changed.

```
$ whoami
fred

$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/fred/bin

$ su

# echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/fred/bin

# whoami
root
```

sudo

The sudo framework exists to delegate to normal users, the ability to run commands as root or as any other user. This delegation is defined inside a configuration file called **/etc/sudoers** and it is advised not to directly edit this file, but to instead use the command **visudo** as sanity checking is part of its codeset.

The basic syntax is a sudo entry is:

“who” “to where”=(can run as these users) “these commands”

Examples:

Inside /etc/sudoers:

```
ricardo     ALL=(ALL)  ALL
```

The user ricardo can log on to ALL hosts and run ALL commands as ALL users.

Inside /etc/sudoers:

```
karl ALL=(oracle, bin, nobody : wheel) NOPASSWD: /usr/bin/whoami, /bin/cat
```

The user **karl** can log onto **ALL** computers which has this entry and run commands with the permissions of the users **oracle**, **bin** and **nobody** and can further elevate his permissions to include that of the group **wheel** and is allowed to run the commands **/usr/bin/whoami** and **/bin/cat**

(As the user **karl**):

```
$ sudo -u nobody -g wheel whoami
nobody

$ ll /var/tmp/foo
----rwx---. 1 fred wheel 12 Jun  4 14:51 /var/tmp/foo

$ cat /var/tmp/foo
cat: /var/tmp/foo: Permission denied

$ sudo -g wheel cat /var/tmp/foo
hello world
```

Inside /etc/sudoers:

```
%wheel      ALL=(ALL)          NOPASSWD: ALL
```

Any member of the group **wheel** can log onto **ALL** computers which has this configuration and run **ALL** commands as **ALL** users without having to quote their password first.

There may be times when you want to restrict the commands or how users run commands which they have been delegated. Consider that the user **ron** has been delegated commands via an entry in /etc/sudoers as follows:

```
ron          ALL=(ALL)  NOPASSWD: /usr/bin/passwd
```

ron is our password reset guy and his job is to be able to change user's passwords.

The big question we need to ask ourselves is, how can this user hurt the system?

Given the configuration and question the answer is that ron can merely reset root's password with the command:

```
$ passwd root
passwd: Only root can specify a user name.
$ sudo passwd root
Changing password for user root.
New password:
BAD PASSWORD: it is based on a dictionary word
BAD PASSWORD: is too simple
Retype new password:
passwd: all authentication tokens updated successfully.
```

Let's fix that flaw! Any command which you want to prevent from being run may be preceded with a !

So a more secure entry in **/etc/sudoers** for ron would be:

```
ron  ALL=(ALL)   NOPASSWD: /usr/bin/passwd [A-z]*, !/usr/bin/passwd *
root, !/usr/bin/passwd root
```

This means that **ron** has to use the **passwd** command and specify at least a username, he also can't use the **passwd** command with or without an argument against the root user account.

To simplify matters, one can create groupings of **USERS**, **HOSTS** and **COMMANDS** by using the parameters: **User_Alias**, **Host_Alias** and **Cmnd_Alias** respectively in **/etc/sudoers**

```
User_Alias WEBMASTERS = shawn, adele, michael

Cmnd_Alias WEBCOM = /sbin/service httpd *, /usr/bin/nano /etc/httpd/*, /usr/bin/nano
/etc/sysconfig/httpd, /bin/cp * /va/r/www/html, /usr/bin/rsync * /var/www/html
```

Above we've created a grouping called **WEBMASTERS** which consists of the users **shawn**, **adele** and **michael**. Note that this grouping is only valid within the context of sudo.

A command alias exists called **WEBCOM** consisting of the command list separated with commas.

To create a sudo entry delegating the grouping called **WEBMASTERS** the ability to run **WEBCOM** and root we create an entry in **/etc/sudoers** as follows:

```
WEBMASTERS ALL=(root) NOPASSWD: WEBCOM
```

SUDO Lab

Let's give the user **betty** the ability to use any command as the user root using **sudo**

Open a terminal and ensure that you are currently working as root. Now run the command **visudo**

Use a **/** to begin searching for the word **wheel** and uncomment the entry for the third match.

You should have a line that reads as follows:
%wheel ALL=(ALL) NOPASSWD: ALL

Save your changes and return to your terminal.

As the user betty is a member of the group wheel she would be able to use any command as any user via sudo and won't be re-prompted for her own password.

Let's try it by using the command **su - betty**

Now that we're logged on as betty let's test our sudo configuration.

Under normal circumstances, the user betty won't be able to see the contents of the file /etc/shadow so let's test this by running **cat /etc/shadow**

Your should get an error stating that permission is denied.

Now run **sudo cat /etc/shadow** and this will cause the command to run as the user **root**.

Let's do something similar now for the user fred but you're going to follow similar steps on your own to devise an entry that will allow fred to run the following commands as root.

useradd
usermod

userdel
groupadd
groupdel
groupmod
passwd
chage

Remember that you have to ask yourself how fred can compromise the system and create the exceptions as necessary.

ISCSI

ISCSI allows us to make use of remote storage units (called targets) over a TCP/IP network and have it seen as localized SCSI storage. To connect to an iSCSI target you need the **iscsi-initiator-utils** package – already installed for you.

Documentation makes discovering and using iSCSI devices simple. The following examples are provided in the manual page for the **iscsiadm** command (man iscsiadm):

Discover targets at a given IP address:

```
iscsiadm --mode discoverydb --type sendtargets --portal 192.168.1.10 --discover
```

Login, must use a node record id found by the discovery:

```
iscsiadm --mode node --targetname iqn.2001-05.com.doe:test --portal 192.168.1.1:3260 --login
```

Logout:

```
iscsiadm --mode node --targetname iqn.2001-05.com.doe:test --portal 192.168.1.1:3260 --logout
```

Once you have logged into an iscsi target it will function as a local disk. To figure out which device it was assigned simply check the output of **dmesg**. Lets assume it was assigned /dev/sdb for this example. You partition and format the target the same way you would a local disk.

!!! Note !!!

When you put the entry in /etc/fstab with the only difference of adding the

_netdev mount option in place of defaults. This lets the kernel know that this drive cannot load until network connection is established.

Managing Storage

The basics

Before we can store data on a disk, we need to make sure that it has been divided up into units called partitions because only partitions can be allocated filesystems and we use filesystems to organize our data.

The flow:

DISK → PARTITION → FILESYSTEM → MOUNT

Disk Management (Physical Partitions)

Each disk has enough entries for 4 partitions in the partition table. The partition table itself is an element of the Master Boot Record which is responsible for loading an operating system off a disk.

partition 1 partition 2 partition 3 partition 4

Once you've exhausted the entries in the main partition table, then you cannot create any more partitions.

Here's a quick breakdown of the different partition types and their purposes.

Primary partitions:

These are the most overrated (and therefore the most overused) partition types.

Advantages	Disadvantages
Can be allocated a filesystem.	Occupies an entry in the main partition table.
Can be used to boot an operating system .	Limited to 4 primary partitions per disk provided that an extended partition is used (in which case you are limited to 3 primary partitions).

Extended partitions:

Every disk making use of the standard partitioning scheme should use this.

Advantages	Disadvantages
Can host 56 logical partitions (they do not occupy an entry in the main partition table).	Cannot be allocated a filesystem.
	Cannot be used to load an operating system.
	Limited to 1 extended partition per partition table.

Logical partitions:

Advantages	Disadvantages
Can create 56 logical partitions	Cannot be used to load an operating system.
Can be allocated a file system	

Partition ID’s

Partitions are allocated partition identifiers which are used to indicate to the system how the partition is to be used.

By default, fdisk creates partitions using ID **0x83** for Primary and Logical partitions and **0x05** for the Extended partition.

Unless you're making use of Clustered Logical Volumes, you do not need to explicitly allocate the partition ID **0x8e** to a Physical Volume.

Typically these partition ID's are interpreted by your OS and may not be used at all.

Disk management tools

Open source means that we have choice! The list below is a set of standard partitioning tools which we have available from the command line alone.

fdisk

This is the most basic of tools which you could use to create partitions and is available on all enterprise Linux operating systems.

Examples:

```
# fdisk
```

This enters the fdisk tool on the only disk installed on this system.

```
# fdisk -l
```

This displays the partition table of all disks connected to the system. To conduct the transaction against a specific disk, refer to the disk in the last argument as **fdisk -l /dev/sdb**

```
# fdisk /dev/sdb
```

This enters the fdisk tool for partitioning the disk represented by the device file /dev/sdb

Once in the fdisk tool you may use the **m** command to get additional help.

```
# fdisk /dev/sdb

Command (m for help): n
Command action
 e   extended
 p   primary partition (1-4)
e
Partition number (1-4): 4
First sector (2048-2097151, default 2048): <hit enter>
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-2097151, default 2097151): <hit
enter>
Using default value 2097151

Command (m for help):
```

!!! NOTE !!!
Selecting <enter> when prompted for a value for the first sector and for the last sector uses the next available sectors as specified in brackets. By doing this we're selecting all the available space to allocate to the extended partition which is GOOD practise. Typically we do not create extended partitions of a fixed size.

In the above example (the commands issued are in bold and highlighted in red) we have executed **n** create a new partition of the type **e** for extended and allocated it slot number **4** in the partition table. After which, the prompt queried which sector I'd like to allocate as the beginning of the partition in which case we just hit enter and then for the last sector we hit enter once more to use the default values which are in brackets. What this essentially has done is dedicate all space on the disk /dev/sdb to create an extended partition.

!!! NOTE !!!

No changes have been committed to the disk at this stage. This only happens when we save the changes using the w command.

Now we will create a logical partition of a fixed size of 100M inside of the extended partition and is continued from the block above:

```
Command (m for help): n
Command action
 l   logical (5 or over)
 p   primary partition (1-4)
l
First sector (4096-2097151, default 4096): <hit enter>
Using default value 4096
Last sector, +sectors or +size{K,M,G} (4096-2097151, default 2097151): +100M
```

So here we are creating a new partition using the **n** command of the type **l** for logical partition. When we're prompted for a value for the first sector we merely hit enter to select the next available sector (as opposed to specifying a specific sector). With the value for the last sector we specify a value here as we want to create a fixed size of 100M using the format +numberunit (as opposed to using all available space inside of the extended partition to allocate to this logical partition) and the command **+100M** achieves this.

The block below is a continuation from the one above:

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

We are able to repeat this process of creating logical partitions as long as we have enough sectors. Once we are done we can commit the changes using the **w** command.

!!! NOTE !!!

If you conduct a partition transaction against the disk that contains your currently running kernel then you will have to reboot in order to generate the device files representing those newly created partitions. You could bypass this process by using the command partx -a /dev/<device file of disk> but the reboot command is what is officially supported.

cfdisk

This is an interface driving variant of disk using the ncurses library. It may not be installed by default but it is available on the distributions focused on in this course.

parted

This command driven partitioning tool supports the IBM partitioning framework as well as
GPT based partitions. GPT based partitions are supported by many operating systems and use Globally Unique Identifiers (GUID's) to identify partitions and allow us to overcome the limitations imposed by the IBM partitioning framework of 64 partitions. GPT uses the 64 bit disk pointers, which allow for a maximum disk partition size of 9.4 Zetabytes, or 9.4 billion TeraBytes. With GPT, we support 128 primary partitions.

Now why are we mentioning all of this GPT stuff? That's because **parted** supports GPT partitions

Examples:

```
# parted /dev/sdb print
```

This command displays the partition table of the disk identified by the device file /dev/sdb.

```
(parted) mklabel gpt
```

Here we specify that we're not using the standard partitioning type (awkwardly referred to as **msdos**), but instead will be using GPT.

```
(parted) mkpart foo 0 1G
```

This command creates a partition with the name foo starting at the first available sector and ending at 1 Gigabyte.

```
(parted) mkpart bar 1G 4G
```

Now we're just creating another partition starting at the first available sector after the previously created partition and ending at 4 Gigabytes.

Just so that we're clear that we're making use of GPT, we're going to list the partition table of **/dev/sdb** using the command **fdisk -cul /dev/sdb**

```
# fdisk -l /dev/sdb

WARNING: GPT (GUID Partition Table) detected on '/dev/sdb'! The util fdisk doesn't support GPT. Use GNU Parted.
---truncated---
```

mkfs

At this stage we haven't created file systems on either partitions created with the utilities fdisk or parted so we have no mechanism to organize files. To create a filesystem on a partition we make use of the mkfs command. With Linux we always have choice, and we support a number of filesystems but for the demonstration we're going to focus on a great performing, journaling file system called ext4.

```
# mkfs -t ext4 /dev/sdb1
```

This command creates a filesystem of the type ext4 using the -t option on the partition represented by the device file /dev/sdb1

Once we have a filesystem allocated to the partitions we wish to use then we can access it by mounting it. Think of the command **mount** as being a verb (in other words, a doing word) which means 'to make available'. So the mount command 'makes a filesystem available' But where?

This is where another term comes in called a **mountpoint** which you should see as a noun (something you can see, feel and touch). A mountpoint is essentially a directory where a filesystem is accessed.

So given that we want to access our newly created ext4 filesystem (which was created on a partition represented by the device file /dev/sdb1) at the mountpoint /data.

We should create the directory /data using the command:

```
# mkdir /data
```

Then we do a non-persistent (it won't survive a reboot) mount:

```
# mount /dev/sdb1 /data
```

You're able to verify that the mount of the filesystem was successful by simply typing:

```
# mount

---truncated---
/dev/sdb1 on /data type ext4 (rw)
```

!!! NOTE !!!

The opposite of mount is **umount** - NOT unmount.

Persistency in Linux is defined in configuration files and file responsible for the persistent mounting of filesystems is **/etc/fstab**

This is a whitespace (1 of more space characters) separated file consisting of 6 fields.

```
|The device file/label/UUID which contains the filesystem you want to mount
|
|
|               |The directory where the filesystem gets mounted to
|               |
|               |
|               |              |The filesystem type
|               |              |
|               |              |
|               |              |             |Options used for mounting
|               |              |             |
|               |              |             |
|               |              |             |              |Do we backup this filesystem using
|               |              |             |              |
|               |              |             |              |
|               |              |             |              |              |The order in which we run fsck
|               |              |             |              |              |
|               |              |             |              |              |
|               |                 |                |              |              |
|               |              |             |              |                 |
/dev/sdb1     /dev/data     ext4            defaults             0                  0
```

!!! TIP !!!

To see some of the possible options which could be used in column 4 see **man mount**

The dump program isn't really used, so 0 is an acceptable value for column 5
3 Possible values exist for the 6th column:

0 = do not automatically run fsck on suspect filesystems
1 = check this filesystem in parallel with the root filesystem

2 = check this filesystem after the root filesystem

!!! NOTE !!!

fsck is automatically called for filesystems which were not unmounted cleanly.

Using UUID's

If you're using local disks then it is acceptable to refer to the device file in /etc/fstab, however if you're using iSCSI it is recommended to use the UUID of a filesystem.

To determine the UUID of the filesystem on /dev/sdb1 use the command:

```
# blkid /dev/sdb1

/dev/sdb1: UUID="b16241e1-df59-4750-95c8-fdb8a74a2f05" TYPE="ext4"
```

The UUID is a randomly generated alphanumeric string which is more accurately identifies a filesystem than a device file.

The line in /etc/fstab using the UUID would look as follows:

```
UUID="b16241e1-df59-4750-95c8-fdb8a74a2f05"    /dev/data     ext4    defaults       0
            0
```

Using Labels instead of UUID's

Labels fail to accurately reference a filesystem uniquely as they may be duplicated and as such should not be used.

Setting a label can be done during filesystem creation time:

```
# mkfs -L MYLABEL /dev/sdb1
```

Or post filesystem creation time:

```
# tune2fs -L MYLABEL /dev/sdb1
```

Should you really wish to mount a filesystem by its label in /etc/fstab then device an entry similar to this:

```
LABEL=DATA     /dev/data     ext4     defaults     0          0
```

Swap Space

Linux divides its physical RAM into chunks of memory called pages. Swapping is the process whereby a page of memory is copied to the preconfigured space on the hard disk, called swap space, to free up that page of memory. The combined sizes of the physical memory and the swap space is the amount of virtual memory available.

Swapping is necessary for two important reasons. First, when the system requires more memory than is physically available, the kernel swaps out less used pages and gives memory to the current application process that needs the memory immediately. Second, a significant number of the pages used by an application during its startup phase may only be used for initialization and then never used again. The system can swap out those pages and free the memory for other applications or even for the disk cache.

How much swap space you need is a very open ended question and to answer that question we monitor the frequency of swapping. Remember, that if your system is performing many swap operations it could mean that you're running out of RAM, in which case you may want to purchase additional RAM units. Also it could mean that you have a buggy application causing memory leaks in which case you will have to get your developers onto it.

```
(parted) mkpartfs swap1 linux-swap 250M 500M
(parted) quit
```

Remember that if you conducted this partitioning transaction against the disk that holds your currently running kernel, then you'd have to reboot in order for the new partition table to be implemented. You may use **partx -a /dev/*<device file of the entire disk>*** as an UNSUPPORTED workaround.

```
# mkswap /dev/sdb6
Setting up swapspace version 1, size = 244732 KiB
no label, UUID=f903bc72-ea5e-4436-97c9-0a19894647d5
```

Now we've allocated the swap filesystem to the partition /dev/sdb6 which means that the partition now has a mechanism to organize swap data.

```
# swapon -s
Filename                        Type            Size            Used    Priority
/dev/dm-1                       partition       2064376         0       -1
```

This command shows us the current swap space utilization and as you can see, our device file **/dev/sdb6** is not listed as being activated. So let's activate it for use already!

```
# swapon /dev/sdb6
```

This command simply activates /dev/sdb6 for swapping and we can verify that it is activated using the command below.

```
# swapon -s
Filename                        Type            Size    Used    Priority
/dev/dm-1                       partition       2064376 0       -1
/dev/sdb6                       partition       244728  0       -2
```

Your swap partition still needs to be activated persistently and we remember that persistency is defined in configuration files. The configuration file we need to edit is **/etc/fstab** by adding the following line:

```
/dev/sdb6     swap          swap          defaults      0       0
```

!!! NOTE !!!

swap partitions are not mounted to a directory, they just get mounted to **swap**

!!! REMEMBER !!!

/etc/fstab requires a whitespace to separate fields. Whitespaces are 1 or more space characters.

!!! TIP !!!

Consider implementing swap space in the form of a logical volume (covered in the next section) as you are able to extend or reduce the logical volume as needed.

Managing Storage Labs

Open a terminal and list the partitions being used on your disk

Device file	Size	Mount point

Make sure that you are logged in as root and create a new extended partition which uses all the remaining space on your disk.

fdisk /dev/sda

Enter n to create a new partition

Now enter e to create an extended partition

Hit enter twice to allocate all the remaining space to the extended partition

Now we will create a logical partition inside of the extended partition which is 100MiB in size.

Enter n to create a new partition

Then enter l to create a logical partition

Now hit enter ONCE

Type +100M to create a partition of 100MiB

Hit p to print the new partition table and verify that all is well

Enter w to save the changes

If you got a warning about the kernel using the old partition table then either **reboot** or run **partx -a /dev/VARIABLE**

Now create the EXT4 filesystem on the newly created logical partition using the command below:

mkfs -t /dev/VARIABLE

If all is well let's then create a mountpoint for the filesystem:

mkdir /data

Before we continue, let's determine the UUID of the filesystem using the command below:

blkid /dev/VARIABLE

Let's create an entry in /etc/fstab for persistent mounting so use your favorite editor to open the file and add a line similar to below:

UUID **/data ext4 defaults 0 0**

Naturally, replace UUID with whatever the actual UUID is of your filesystem.

Let’s reprocess /etc/fstab by running **mount -a**. If all went well there should be no errors via STDOUT

Verify that your filesystem is mounted to /data by running **mount** without any options.

Now with minimal assistance and drawing inspiration from above, create a swap partition of 256MiB in size

Logical Volume Management

This is the recommended way to manage storage in the enterprise as it allows for growth and flexibility in your filesystem management.

3 components make up the LVM framework: Physical Volumes, Volume Groups and Logical Volumes.

> **!!! NOTE !!!**
>
> Do not confuse Logical Partitions and Logical Volumes, these are separate entities.

Physical Volumes

These are partitions or entire disks which have been identified as being part of the LVM framework.

To create an entire disk as a PV, you may use the command **pvcreate**. If your goal is to initialize only a portion of a disk to use as a PV then you need to create a partition and initialize the device file representing your partition as a PV.

```
# pvcreate /dev/sdc
```

```
Writing physical volume data to disk "/dev/sdc"
Physical volume "/dev/sdc" successfully created
```

We can view all PV's using the command **pvs** or to get slightly more detail use **pvscan**

```
# pvs
PV      VG      Fmt  Attr PSize PFree
/dev/sda2  vg_gpg1 lvm2 a--  5.37g      0
/dev/sdc       lvm2 a--  1.00g 1.00g
```

or

```
# pvscan
PV /dev/sda2   VG vg_gpg1        lvm2 [5.37 GiB / 0   free]
PV /dev/sdc                      lvm2 [1.00 GiB]
Total: 2 [6.37 GiB] / in use: 1 [5.37 GiB] / in no VG: 1 [1.00 GiB]
```

!!! TIP !!!

All the PV related commands are prefixed with pv so to see all the PV related commands we type in pv<TAB><TAB>.

Volume Groups

A grouping of PV's is called a Volume Group and this collective entity is given a name. We like prefixing our VG's using **VG_** but this is not a requirement. Why we do this is to easily identify which items are VG's and we do so in uppercase so that they stand out.

To create a VG we need to specify at least 1 PV to add to it, so to create a VG called VG_DB and add the PV /dev/sdc to it, we could use:

```
# vgcreate VG_DB /dev/sdc
Volume group "VG_DB" successfully created
```

On creation of a VG, a device file is created called **/dev/<*VG NAME*>**

Now to see all VG's on our system we could use the command **vgs**:

```
# vgs
VG          #PV  #LV  #SN  Attr      VSize      VFree
VG_DB       1    0    0    wz--n-    1020.00m   1020.00m
vg_gpg1     1    2    0    wz--n-    5.37g      0
```

To view the properties of a newly created VG_DB we use the command **vgdisplay:**

```
# vgdisplay VG_DB
--- Volume group ---
VG Name               VG_DB
System ID
Format                lvm2
Metadata Areas        1
Metadata Sequence No  1
VG Access             read/write
VG Status             resizable
MAX LV                0
Cur LV                0
Open LV               0
Max PV                0
Cur PV                1
Act PV                1
VG Size               1020.00 MiB
PE Size               4.00 MiB
Total PE              255
Alloc PE / Size       0 / 0
Free  PE / Size       255 / 1020.00 MiB
VG UUID               OV2mj7-1MBm-VVTV-CF1M-t5LP-ncsO-Xy4rQ3
```

As you can see, we have not yet created any Logical Volumes out of the space provided by this Volume Group.

This indicates that we have 1 PV added to this VG so far, we of course are able to add more.

1020 MiB is the total amount of space in MiB which is available for us to use to allocate to Logical Volumes.

Inside the LVM framework, data cannot be organized into cylinders (as when you store data on a filesystem on a single hard drive). Because our VG may consist of multiple PV (and therefore multiple hard drives) we need another way to organize data and this is done by physical extents. Each physical extent by default is 4 MiB and may be changed at VG creation time using the -s option.

We have 255 extents which make up our VG given the PV's added to it. As you can now see (Total PE) * (PE Size) = (VG Size)

This shows us how many PE have been allocated as well as the size in MiB that it represents.

Finally we can see how many free PE we have as well as the size in MiB that it represents.

Now that we have a VG, we are able to create as many Logical Volumes as we like, as long as we have sufficient space (identified by extents) inside the VG.

!!! TIP !!!

As with the PV commands, all the VG related commands are prefixed with vg so to see all the VG related commands we type in lv<TAB><TAB>.

Logical Volumes

This is what we've been building up to, a flexible storage subsystem which can be allocated a filesystem of your choosing. LV's occupy extents (space) of a VG. So you can create a LV of less than or equal to the amount of free PE as viewed using the command **vgdisplay VG_DB**

Our objective now is to create a LV which uses 20 Physical extents of the VG called VG_DB (each PE is 4MiB in size so that means our LV will be 80MiB in size) and name it LV_PGSQL. We like prefixing our LV's with an uppercase LV_ so that we could easily separate the multiple elements that make up our LVM framework.

```
# lvcreate -n LV_PGSQL -l 20 VG_DB
```

To see all LV's inside VG_DB we use:

```
# lvs VG_DB
LV      VG          Attr       LSize  Pool Origin Data%  Move Log Copy%
Convert
LV_     PGSQL       VG_DB      -wi-a--- 80.00m
```

To view detailed information about LV_PGSQL which is inside VG_DB we use the command:

```
# lvdisplay VG_DB/LV_PGSQL
--- Logical volume ---
LV Path                /dev/VG_DB/LV_PGSQL
LV Name                LV_PGSQL
VG Name                VG_DB
LV UUID                C93cDe-3Ew9-Nb5O-Moqj-9Uji-Cl1w-l9qNNa
LV Write Access        read/write
LV Creation host, time elsa.enterpriselinux.pro, 2013-02-26 21:00:36 -0700
LV Status              available
# open                 0
LV Size                80.00 MiB
Current LE             20
Segments               1
```

```
Allocation              inherit
Read ahead sectors      auto
- currently set to      256
Block device            253:2
```

The device file which represents the Logical Volume itself.

The size of the LV is 80MiB

This represents the number of Logical Extents which make up our LV (each LE is equal in size to a PE which is a property of the VG and was left at the default of 4MiB).

Segments indicate the number of PV which contributes extents to this LV.
Before we can use this LV for storage we need to allocate it a filesystem using the **mkfs** command:

```
# mkfs -t ext4 /dev/VG_DB/LV_PGSQL
mke2fs 1.41.12 (17-May-2010)
---truncated---
```

Now let's create a mountpoint for this filesystem:

```
# mkdir -p /db/pgsql
```

And finally, let's make sure that this filesystem is persistently mounted to /db/pgsql by adding the following entry to **/etc/fstab:**

```
/dev/VG_DB/LV_PGSQL  /db/pgsql          ext4  defaults     0 0
```

Let's reprocess /etc/fstab to make sure that our entry is correct and that the filesystem is mounted accordingly:

```
# mount -a
```

!!! NOTE !!!

No feedback when running **mount -a** is a good thing! If there was some form of feedback then it means that something was not correctly processed inside **/etc/fstab**

Adding More Disks

Remember that your LV can only be as big as the number of free Physical Extents which make up your VG. If you're running out of space and wish to add more PE, you may add additional Physical Volumes to a Volume Group using the **vgextend** command. This is a 2 step process.

1. Initialize storage as a PV using pvcreate
2. Add newly initialized PV to the existing VG

Step 1:

```
# pvcreate /dev/sdd
Writing physical volume data to disk "/dev/sdd"
Physical volume "/dev/sdd" successfully created
```

Step 2:

```
# vgextend VG_DB /dev/sdd
Volume group "VG_DB" successfully extended
```

Now when we run the **vgs** command against VG_DB we see that there are 2 PV which make up this VG.

```
# vgs VG_DB
VG      #PV #LV #SN Attr   VSize VFree
VG_DB   2   1   0 wz--n- 1.99g 1.91g
```

Growing a Logical Volume

This is a common task for many system administrators and can even be completed online.
That's right, you do not have to unmount your filesystem and you users may continue working
as this is a completely transparent process.

Recall that our VG_DB/LV_PGSQL is 80 MiB or 20 extents in size and we will now grow it to be 160 MiB (40 extents).

```
# lvresize -r -L 160M /dev/VG_DB/LV_PGSQL
```

This could also be achieved by running executing the following 2 step process:

Step 1:

```
# lvextend -L 160M /dev/VG_DB/LV_PGSQL
Extending logical volume LV_PGSQL to 160.00 MiB
Logical volume LV_PGSQL successfully resized
```

Step 2:

```
# resize2fs /dev/VG_DB/LV_PGSQL
resize2fs 1.41.12 (17-May-2010)
Filesystem at /dev/VG_DB/LV_PGSQL is mounted on /db/pgsql; on-line resizing required
old desc_blocks = 1, new_desc_blocks = 1
Performing an on-line resize of /dev/VG_DB/LV_PGSQL to 163840 (1k) blocks.
The filesystem on /dev/VG_DB/LV_PGSQL is now 163840 blocks long.
```

Now let's verify that the VG_DB/LV_PGSQL is in fact 160 MiB:

```
# lvdisplay VG_DB/LV_PGSQL
--- Logical volume ---
LV Path                /dev/VG_DB/LV_PGSQL
LV Name                LV_PGSQL
VG Name                VG_DB
LV UUID                C93cDe-3Ew9-Nb5O-Moqj-9Uji-Cl1w-l9qNNa
```

```
LV Write Access         read/write
LV Creation host, time elsa.enterpriselinux.pro, 2013-02-26 21:00:36 -0700
LV Status               available
# open                  1
LV Size                 160.00 MiB
Current LE              40
Segments                1
```

Reducing a Logical Volume

This is quite a rare task for systems administrators but if it needs to be performed, it is imperative that you know that this is an OFFLINE process. Of course, you can only reduce the LV to as much as what is used.

Again we are able to use the **lvresize** command:

```
# lvresize -r -L 20M /dev/VG_DB/LV_PGSQL
```

We could also execute the individual steps which the **lvresize** command performs:

1. Unmount the filesystem
2. Verify that the filesystem is clean before reducing its size
3. Reduce the filesystem size
4. Reduce the Logical Volume size

Our VG_DB/LV_PGSQL is currently 160 MiB in size and we want to take it down to 20 MiB.

Step 1:

```
# umount /dev/VG_DB/LV_PGSQL
```

This accomplishes our first step which is to take the LV offline so that it's not in use.

Step 2:

```
# fsck -f /dev/VG_DB/LV_PGSQL
fsck from util-linux-ng 2.17.2
e2fsck 1.41.12 (17-May-2010)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
```

```
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/mapper/VG_DB-LV_PGSQL: 11/40960 files (9.1% non-contiguous), 10819/163840
blocks
```

Now that we've verified that the filesystem is clean and is error free we may proceed.

Step 3:

```
# resize2fs /dev/VG_DB/LV_PGSQL 40M
resize2fs 1.41.12 (17-May-2010)
Resizing the filesystem on /dev/VG_DB/LV_PGSQL to 40960 (1k) blocks.
The filesystem on /dev/VG_DB/LV_PGSQL is now 40960 blocks long.
```

We're almost there! Our LV has had the filesystem reduced to 40 MiB so the final step is to reduce the LV size.

Step 4:

```
# lvreduce -L 40M /dev/VG_DB/LV_PGSQL
WARNING: Reducing active logical volume to 40.00 MiB
THIS MAY DESTROY YOUR DATA (filesystem etc.)
Do you really want to reduce LV_PGSQL? [y/n]: y
Reducing logical volume LV_PGSQL to 40.00 MiB
Logical volume LV_PGSQL successfully resized
```

The real success is to insure that the filesystem is available when it is remounted, so let's go ahead and do that.

```
# mount -a
```

Remember the tip earlier? If the **mount -a** runs error free, then all was good. But just in case you're paranoid, here's a final verification:

```
# lvdisplay VG_DB/LV_PGSQL
```

```
--- Logical volume ---
LV Path                /dev/VG_DB/LV_PGSQL
LV Name                LV_PGSQL
VG Name                VG_DB
LV UUID                C93cDe-3Ew9-Nb5O-Moqj-9Uji-Cl1w-l9qNNa
LV Write Access        read/write
LV Creation host, time elsa.enterpriselinux.pro, 2013-02-26 21:00:36 -0700
LV Status              available
# open                 1
LV Size                40.00 MiB
Current LE             10
Segments               1
---truncated---
```

Removing a disk from the LVM framework

Firstly, we need to make sure that any data on the PV which we want to remove (in our case this is /dev/sdc) is relocated to another PV. We may achieve this by using the command:

```
# pvmove /dev/sdc /dev/sdd
/dev/sdc: Moved: 100.0%
```

Now that the data has been moved, we may reduce the LV by removing the PV from the VG:

```
# vgreduce VG_DB /dev/sdc
(no output)
```

At this stage, the PV /dev/sdc is still part of the LVM framework so we can remove it by using the command:

```
# pvremove /dev/sdc
(no output)
```

!!! TIP !!!

The trend continues! So to see all the LV related commands, remember that they're all prefixed with lv, type in lv<TAB><TAB>.

Logical Volume Management Labs

Now we will create a logical volume called LV_WEB out of a volume group called VG_SERVICES. The volume group is to have 1 physical volume of 500MiB added to it with an extent size of 16MiB. The logical volume LV_WEB must use 5 extents.

Create a partition (preferably a logical partition of 500MiB) using **fdisk**

Run **pvcreate /dev/device_file_of_logical_partition.** This creates turns your newly created partition into a physical volume and may be verified with the command **pvs**

Now let's create a new volume group out of the physical volume which has an extent size of 16MiB with the command **vgcreate -s 16M VG_SERVICES /dev/device_file_of_phyiscal_partition**

Verify that the volume group has been added by running **vgs**

Additionally, verify the properties (like the extent size) of the volume group by running **vgdisplay VG_SERVICES**
Finally create the logical volume with the following command:

lvcreate -l 5 -n LV_WEB VG_SERVICES

Verify that the logical volume is created using the commands **lvs** and **lvdisplay /dev/VG_SERVICES/LV_WEB**

Before we can use that logical volume for storage we need to give it a filesystem and mount it.

Use the EXT4 filesystem and mount it persistently to /var/www

Using inspiration from above, remove the swap space you've previously created and implement a logical volume for swapping called LV_SWAP which uses 10 extents from the volume group VG_SERVICES.

Cool filesystem tools

blkid

Determines the UUID of devices so that you can accurately identify a filesystem.

Example:

```
# blkid /dev/sda1
/dev/sda1: UUID="c9fb705d-19eb-4493-ab72-0eafceb34045" TYPE="ext4"
```

lsblk

This is a real gem of a command and is new to EL6. It lists all your block devices in tree view along with their respective mount points, filesystem types and labels.

Example:

```
# lsblk -f
NAME                    FSTYPE          LABEL MOUNTPOINT
sr0
```

```
sda
├─sda1                  ext4                        /boot
└─sda2                  LVM2_member
  ├─VolGroup-lv_root (dm-0) ext4                      /
  └─VolGroup-lv_swap (dm-1) swap                      [SWAP]
vda                                 LVM2_member
├─vda1
├─vda5
├─vda6
├─vda7
├─vda8
├─vda9
├─vda10
├─vda11
├─vda12
├─vda13
├─vda14
├─vda15
└─VG_DATA-LV_PRIV (dm-2)
  └─private (dm-3)      ext2    FOO   /excalibur
```

lsscsi

Lists information about currently connected SCSI devices.

findmnt

Another awesome command which will display your mounted filesystems and their properties.

tune2fs

This is used to manipulate the properties of an EXT based filesystem. With it one can allocate a label to the filesystem, implement default mount options (thereby eliminating the need to explicitly specify the mount option on each server's /etc/fstab file), set the frequency of how often fsck checks your filesystem, specify journal options and much more!

Examples:

Let’s change a the label of /dev/sda1

```
# tune2fs BOOT /dev/sda1
```

Now let’s set the acl mount option as a default for /dev/sda4

```
# tune2fs -o acl /dev/sda4
```

Similarly, to remove a mount option put a caret (^) in front of that option as follows:

```
# tune2fs -o ^user_xattr /dev/sda4
```

To force fsck (e2fsck) to check the /dev/sda4 filesystem every 20 mounts:

```
# fsck -c 20 /dev/sda4
```

Normal POSIX Permissions

Regular (POSIX) Permissions

Permissions determine access to files and directories and are really simple!

Everything in computing is identified numerically (like users are not identified by their usernames but by their user ID's). The identifier used for files is called an index node which we typically refer to as an inode. Every file has an inode and the inode is essentially metadata and is used to store data & time stamps, ownership, permissions and pointers to the data blocks for the file on the filesystem.

To determine the permissions associated with a file we use the command **ls -l** which displays a long listing of the properties of a file as it is stored in the inode.

```
# ls -l /etc/fstab
-rw-r--r--. 1 root root 831 Mar  8 12:15 /etc/fstab
```

To display the permissions of a directory you would use **ls -ld**. Using ls -l without the -d would do a long listing of the contents of the directory instead.

```
# ls -ld /etc
drwxr-xr-x. 67 root root 4096 May 18 19:09 /etc
```

!!! TIP !!!
To view the inode associated with a file use the -i option with the ls -l command

Permissions make up 10 bits and they are first item on the left hand side which reads:
-rw-r--r--.

The first bit tells us the type of file it is, a **-** indicates that it is a normal file.

Value of first bit	**Meaning**
-	normal file
d	directory
l	symbolic link
c	character file
b	block file
p	fifo
s	socket

The next 9 bits read **rw-r--r--** and these are the permissions for access control.

Permissions are easy as we have 3 access permissions

r - read
w - write

x - execute

These permissions are always listed in that order and may be allocated to 3 entities called:

- the owning user (the user who is associated with the file)
- the owning group (the group which is associated with the file)
- others (everyone who isn't the owning user or isn't a member of the owning group)

Permissions are processed in that order and we stop at the first match.

Given the output of the following command

```
# ls -l /etc/fstab:
-rw-r--r--. 1 root root 831 Mar  8 12:15 /etc/fstab
```

1. /etc/fstab is a normal file
2. the owning user has rw-
3. members of the owning group have r--
4. others have r--
5. the owning user is root
6. and the owning group is root

If I am logged on as the user fred, Linux determines your access as follows:

1. Is the currently logged on user the owning user of the file?
    If yes then the user has the permissions associated with the owning user and stop processing.
    If no then process the next element

2. Is the currently logged on user a member of the owning group of the file?
    If yes then the user has the permissions set for the owning group and stop processing.
    If no then process the next element

3. You're qualified by others

The user fred therefore has r-- permissions to the file /etc/fstab

Another example:

```
# ls -l /var/tmp/foo
-rw----rwx. 1 fred wheel 0 Mar  8 12:15 /var/tmp/foo
```

fred is a member of the groups fred and wheel
wilma is a member of the groups wilma and wheel
barney is a member of the group barney

Following the rules above:

fred has rw (match to the owning user)
wilma has --- (match to the owning group)
barney has rwx (the user is neither the owning user nor a member of the owning group and is qualified by others)

Permissions impact files differently to directories

For files:

If you want to	Then you need
See the contents of a file	r--
Run an executable	r-x
Change the contents of a file	rw-

For directories:

If you want to	Then you need
Change to a directory	r-x
Do a directory listing	r-x
Create a file in a directory	rw-
Delete a file from a directory	rw-

Permissions can be changed individually with the **chmod** command by root or the owner of a file. Ownership can only be transferred by root using the command **chown**.

```
# ls -l /var/tmp/bar
-rw-rw-r--. 1 fred fred 0 May 19 09:17 /var/tmp/bar
```

To add (+) x for the owning user,, remove (-) w from the owning group and set (=) the permissions for others to -, the following command would suffice:

```
# chmod u+x,g-w,o= /var/tmp/bar
```

Resulting in:

```
# ls -l /var/tmp/bar
-rwxr-----. 1 fred fred 0 May 19 09:17 /var/tmp/bar
```

Permissions can be expressed numerically, as well. Numbers are assigned to each permission.

Read = 4

Write = 2

Execute = 1

Therefore, the permission set from above (-rwxr-xr--) could be broken down into a three digit number to describe the permission of the user, group and other. The user's permissions would be 4+2+1 (or 7), the group permissions are 4+1 (or 5) and everybody else has 4 (read only). Expressed together, the permissions of this file are 754. If you have a background in web administration you will likely use numbers to set permissions, like this:

```
# chmod 755 file
```

Normal Posix Permissions Labs

We want to ensure that only users from the group wbs have the ability to write to the /data directory and that users are able to delete files which they own. No other users should have access to the /data directory.

1. chown .wbs /data
2. chmod 1770 /data

The first command above makes sure that wbs is the owning group of the /data directory.

The second command allocates the Sticky bit (1) and RWX (7) permissions to the owning user (which should be root) and the owning group.

Make sure that any newly created files in /data are automatically group-owned by the group wbs by using the Set GID bit. Remember to preserve the Sticky bit on the directory.

Special Permissions

So, you think that that's it? Well, we have 3 more permissions for you that are really special!

Set User ID bit

The Set User ID (suid) permission is given to executable files when you want other users to execute a command as the owning user of the file.

As the user ricardo, I will attempt to run the cat command against the file /etc/shadow:

```
$ ll /bin/cat
-rwxr-xr-x. 1 root root 45224 Jun 22  2012 /bin/cat
```

As you can see from the above, the user ricardo is qualified by 'others' which means that he has r-x permissions against the executable /bin/cat

```
$ ll /etc/shadow
----------. 1 root root 1279 Feb 20 01:29 /etc/shadow
```

No one besides root is able to access /etc/shadow so when the user ricardo attempted to access the /etc/shadow file using the executable /bin/cat the following is the result:

```
$ cat /etc/shadow
cat: /etc/shadow: Permission denied
```

And rightfully so!

Now what we can do is allocate the SetUID bit to the executable /usr/bin/tac which means that when a user executes /usr/bin/tac, the executable runs with the permissions of the owning user of the file instead of with the permissions of the user who is attempting to run the executable.

To allocate the SetUID bit we could use:

```
# chmod u+s /usr/bin/tac

or

# chmod 4755 /usr/bin/tac
```

This results in the x permission for the owning user being replaced with an s.

```
# ll /usr/bin/tac
-rwsr-xr-x. 1 root root 93720 Jun 22  2012 /usr/bin/tac
```

So, while the following command remains unsuccessful:

```
# cat /etc/shadow
cat: /etc/shadow: Permission denied
```

The following command works:

```
# tac /etc/shadow
sleepy:!!:15755:0:99999:7:::
---truncated---
```

Set Group ID bit

The Set Group ID bit may be allocated to both files and directories but the behaviour is very different. The Set Group ID bit is largely set on directories but when set on executable files its behavior is similar to the SetUID bit but instead of affecting the owning user, it is applicable to the owning group. (i.e. the executable runs with the permissions of the owning group and now with the permissions of the user attempting to run the executable).

Primarily, the SetGID bit is set on directories to insure that the owning group of newly created files in the directory is the same as the owning group of the directory on which the bit is set. This is primarily done on collaborative directories to insure

that members of a group always have the access they require to newly created files in a directory.

```
# mkdir /common
# ls -ld /common
drwxr-xr-x. 2 root root 4096 Feb 20 02:15 /common
```

The Set Group ID bit may be allocated using 2 ways:

```
# chmod g+s /directory
```

or

```
# chmod 2777 /directory
```

```
# ls -ld /common/
drwxrwsrwx. 2 root root 4096 Feb 20 02:15 /common/
```

Where the x is expected in the permissions for the owning group, it is replaced with an s. Now what we'll do is change the owning group of /common to 3stooges using the command:

```
# chown .3stooges /common
```

Any newly created files which are created in the /common directory will have their owning group automatically set to 3stooges regardless if the user was a member of that group or not.

```
# ls -ld /common
drwxrwsrwx. 2 root 3stooges 4096 Feb 25 11:43 /common
```

Now I'll log on as a different user called ricardo

```
# groups
ricardo
```


See? That user called **ricardo** is only a member of a group called **ricardo**

But …

```
# touch foo
# ls -l
total 0
-rw-rw-r--. 1 ricardo 3stooges 0 Feb 25 11:47 foo
```

Even though the user **ricardo** is not a member of the group **3stooges**, any files created in that directory are automatically group owned by the group **3stooges**.

Sticky Bit

The sticky bit permission protects directory contents from non-owners deleting them. This is fantastic for collaborative directories where you want users to be able to create files in a directory but you don't want others to delete files which they don't necessarily own.

The Sticky bit may be allocated using 2 ways:

```
# chmod o+t /directory

or

# chmod 1777 /directory
```

Let's get started!

```
# chmod 1777 /common/
# ls -ld /common/
drwxrwsrwt. 2 root 3stooges 4096 Feb 25 11:47 /common/
```

Inside the directory **/common**, there is a file which was created earlier by the user **ricardo** called **foo.** Because the user **ricardo** is the owner of the file and the sticky

bit has been set on the directory, it means that no other user except for the owner may delete this file.

With the sticky bit and logged on as the user **karl**:

```
$ ls -ld /common
drwxrwsrwt. 2 root 3stooges 4096 Feb 25 11:47 /common
```

In this command we determine the, via others, the user Karl has rwx to the directory /common which means that this user can create and delete ANY files in this directory. So as **karl**:

```
$ rm -rf /common/foo
rm: cannot remove `/common/foo': Operation not permitted
```

However, the sticky bit has been set on the directory which now means that only the owner of a file (in this case it is the user ricardo), can delete the file. Note the error message, “Permission Denied” means that the user did not have permissions to perform the transaction but in this case, “Operation not permitted” means that the user did have the necessary permissions but the operation was not allowed via the sticky bit.

Without the sticky bit as the user **karl**:

```
$ ls -ld /common/
drwxrwsrwx. 2 root 3stooges 4096 Feb 25 11:47 /common/
```

As you can see, the user **karl** derives his rwx permissions via others. This means that he can create and delete files, even those files which he does not own.

```
$ rm -rf /common/foo
$ ll /common/foo
ls: cannot access /common/foo: No such file or directory
```

See? No error! That’s how the sticky bit works.

Summary

Who	The Sticky bit
What it affects	Directories only
How to set it	chmod o+t /dir or chmod 1777 /dir (you can substitute 777 for any of the standard UGO permissions)
How to identify it	A t will be seen in place of the execute permission for others. If others have not been allocated the execute permission then a T will be shown in place of the execute permission for others
Purpose	Anyone who has write permissions to a directory is able to delete files from that directory. The sticky bit ensures that only owners (who have write permissions to a directory) are able to delete files.
Useful for	Collaborative teaming directories where you want users to be able to create files in a directory but you only want users to delete files which they've created.

Who	The Set Group ID bit
What it affects	Files and Directories
How to set it	chmod g+s /file *or* chmod g+s /dir or chmod 2777 /file *or* chmod 2777 /dir

	(you can substitute 777 for any of the standard UGO permissions)
How to identify it	A s will be seen in place of the execute permission for the owning group. If the owning group have not been allocated the execute permission then a S will be shown in place of the execute permission for the owning group.
Purpose	For files: If the file is an executable, the executable runs with the permissions of the owning group of the file and not with the permissions of the user who is attempting to execute the file. For directories: Any newly created children in that directory will have the same owning group as the owning group of the parent instead of having the owning group set to the primary group of the user who created the file (or directory).
Useful for	Files: When you want files to run with the permissions of another group instead of running with the permissions of the user. Directories: Great for collaborative teaming directories where you want to ensure that other members of a group automatically have access to newly created children of that directory. This is very often used in conjunction with the Sticky bit.

Who	The Set User ID bit
What it affects	Files only
How to set it	chmod u+s /file or chmod 4777 /file

	(you can substitute 777 for any of the standard UGO permissions)
How to identify it	A s will be seen in place of the execute permission for the owning user. If the owning user has not been allocated the execute permission then a S will be shown in place of the execute permission for the owning user.
Purpose	If the file is an executable, the executable runs with the permissions of the owning user of the file and not with the permissions of the user who is attempting to execute the file.
Useful for	Ensuring that a program always runs with the permissions of another user.

ACLs (Access Control Lists)

Sometimes one would require different permissions for different users and groups. The problem is that a file may only have 1 owning user and 1 owning group. So if the requirement is to allocate different permissions for these entities, you could make use of ACL's.

ACL's may be implemented by root or the owning user of a file provided that the filesystem is mounted with the acl option. This is achieved by using the acl option in the options block (column 4) in /etc/fstab but a much better way to do this would be

to implement the acl option as a mount option in the filesystem metadata using the following command:

```
# tune2fs -o acl /dev/sda2
```

Viewing an ACL

Any file which has a + sign to the right of the permissions block has been allocated an ACL which is viewable with the **getfacl** command.

Setting an ACL

The setfacl command can be used to set an ACL. Here is an example of adding and ACL for a user:

```
$ ll
total 0
-rw-rw-r--. 1 ricardo ricardo 0 Feb 20 00:42 foo
```

As you can see, there is no ACL attached to the file called foo.
Our objective is to implement an ACL on the file foo with the following properties:

- ricardo remains the owning user with rw permissions
- wheel is the owning group with r permissions
- others have all their access revoked
- members of the group 3stooges are to have rwx permissions
- members of the group 7dwarves are to have r-x permissions
- the user sleepy who is a member of the group 7dwarves must not have any permissions

To achieve upon this goal we would run the following commands:

```
# chown .wheel foo
```

The above command sets the owning group for the file foo to wheel.

```
# chmod 640 foo
```

The above command sets the permissions to rw- for the owning user, r-- for the owning group and others have no permissions allocated.

```
# setfacl -m g:3stooges:rwx,g:7dwarves:rx,u:sleepy:- foo
```

The final command insures that additionally, members of the group 3stooges are allocated rwx, members of the group 7dwarves are allocated rx and the user sleepy has no permissions.

The result would be:

```
$ ll foo
-rw-rwxr--+ 1 ricardo ricardo 0 Feb 20 00:42 foo
```

```
$ getfacl foo
# file: foo
# owner: ricardo
# group: ricardo
user::rw-
user:sleepy:---
group::rw-
group:3stooges:rwx
group:7dwarves:r-x
mask::rwx
other::r--
```

ACL’s are always processed in the following order and processing is stopped at the first match:

1. owning user
2. named user(s)

3. owning group
4. named group(s)
5. others

So given that the user sleepy is logged in and is a member of the group 7dwarves, we would ask the following questions:

1. Is the user in question the owning user? Our answer is **no**
2. Is the user in question one of the named users? Here we have a match so we do not process any further.

So the permissions which would be effective for sleepy would --- (none)

If you thought that sleepy would derive permissions from his membership through the group 7dwarves then you processed beyond the match we had for the owning user which is not allowed.

Note 1: ACLs are a quick and easy way to solve permission issues, but the more you have the more there are to maintain.

Note 2: ACLs are not implemented in the index node of a file which means that when files are backed up using tools like tar, then the corresponding ACL is not backed up with it. Instead we recommend using the star command to backup your files and associated ACLs.

ACL's and inheritance

When an ACL is set on a directory, any newly created children will not inherit that ACL. To ensure this behavior, one would have to create a separate ACL called a default ACL.

```
# mkdir /data

# setfacl -m
u:fred:rwx,u:betty:rw,g:wheel:rx,g:flintstones:rwx,g:rubbles:rwx /data
```

fred is a member of the group flintstones
wilma is a member of the group flintstones
barney is a member of the group rubbles and wheel
betty is a member of the group rubbles

```
# getfacl /data
getfacl: Removing leading '/' from absolute path names
# file: data
# owner: root
# group: root
user::rwx
user:betty:rw-
user:fred:rwx
group::r-x
group:wheel:r-x
group:flintstones:rwx
group:rubbles:rwx
mask::rwx
other::r-x
```

Let’s determine what permissions the users actually have to the /data directory:

fred has rwx (match to a named user)
wilma has rwx (match to a named group)
barney has rwx (match to multiple named groups in which case the permissions are combined)
betty has rw (match to a named user)

When children are created below the directory /data, those ACL’s are not passed on.

```
# touch foo.1
# ll
total 0
-rw-r--r--. 1 root root 0 May 19 09:56 foo.1
```

There’s no + sign next to the permissions, therefore the ACL was not passed on.

To ensure this one would create a default ACL.

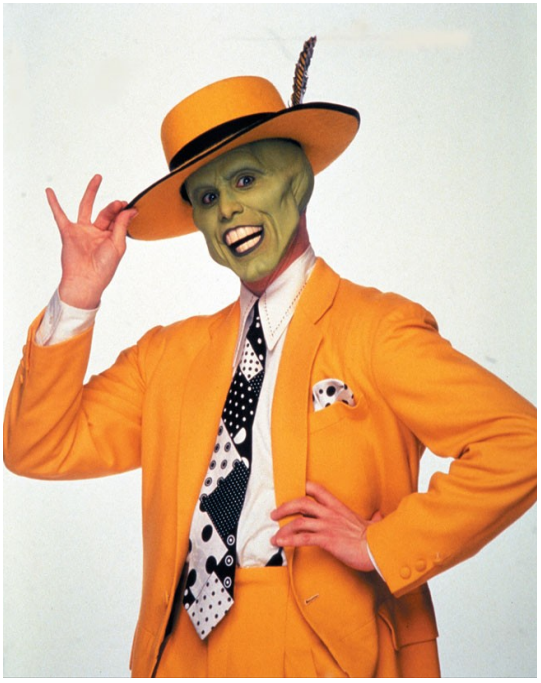
!!! NOTE !!! A default ACL affects newly created children only.

!!! NOTE !!! A default ACL can be different to the actual ACL which is set on a directory.

```
# setfacl -d -m u:fred:rwx,u:dino:- /data
```

This ensures that the user fred always has rwx to newly created children of /data and the user dino always has no permissions to newly children created of the same directory.

The mask



A great way to prevent permissions from being inherited to a child is to implement a mask.

The mask specifies which permissions, should you already have it, you're allowed to use on that file.

Let's say you have apples, bananas and pears and the mask says that you may only eat apples and pears, while you have been given bananas, you may not consume them.

```
# getfacl foo.2
# file: foo.2
# owner: root
```

```
# group: root
user::---
user:fred:rwx                   #effective:rw-
user:dino:---
group::r-x                      #effective:r--
mask::rw-
other::r--
```

The mask above is set to rw- and the user fred has been given rwx. Therefore his effective permissions are rw-

To change the mask for that file, we use the command:

```
# setfacl -m m:rwx /data/foo.2
# getfacl /data/foo.2
getfacl: Removing leading '/' from absolute path names
# file: data/foo.2
# owner: root
# group: root
user::---
user:fred:rwx
user:dino:---
group::r-x
mask::rwx
other::r--
```

As you can now see, the mask allows all permissions which have been allocated.

!!! NOTE !!! Many filesystems support these extended ACL's including NTFS and NFS so one can implement a really robust file server

!!! WARNING !!! Many GUI programs (like Nautilus) do not support and recognize ACL's

ACL Labs

Let’s implement Extended ACL’s on /data

1. Make sure that the /data filesystem supports extended ACL’s and do not use /etc/fstab to implement this.

2. Tweak /data so that any newly created files and directories will be readable and writable for members of the group wheel. Also ensure that the wbs group can create files and directories in /data/wbs.

3. Create a directory called /data/flintstones and make sure that members of the flintstones group have read, write and execute permissions to that directory and any newly created files and directories. Set it so that the user wilma does not have the write permission.

4. The user barney should always be able to write to files below /data

!!! TIP !!!

Check your work to make sure that the above is achievable

Filesystem Layouts

Having a good filesystem layout prevents many headaches as the server matures. While we have previously discussed partitioning, and the use of traditional partitions vs logical volumes, we will now look at which directories should operate as independent filesystems.

Ultimately the role of the server determines the filesystem layout and there is no “one solution fits all”.

Most of the filesystems should be created as Logical Volumes as it allows for the scalability that Linux operating systems are famous for. Logical Volumes can be reduced (not done very often) and extended (this is done very often) as the needs arise.

The basics

These guidelines are applicable to all new servers and should be followed.

- **You need 1 Primary partition of 1GiB for the /boot filesystem.**

The kernel, initramfs and GRUB are stored below /boot and don't occupy much space. While in reality your /boot filesystem will remain small, we create it as 1GiB because resizing a primary partition isn't supported in SUSE Linux.

- **Create a Logical Volume for a swap partition using the square root of the RAM of your server as the size.**

Your server begins to use swap space when you begin to run out of memory. If your system is running out of memory you will use swap space, should you need more swap space you can extend the logical volume which is used for swap. Creating swap space as a regular partition limits you a fixed size and you would have to create additional partitions to provision additional swap space. Not creating a partition for swap is not recommended at all. Remember that if you are running out of RAM it generally means that either your workload could have increased and you need to increase the RAM in your server or it could also mean that you have a buggy application which is causing a memory leak.

- **Create a Logical Volume for /tmp and /var/tmp which is about 5GiB in size**

Most administrators are aware that /tmp is used as temporary space and that any user can use the space below /tmp but many are not aware that we have a second unit of temporary space in the form of /var/tmp. The directories differ based on their purging policies (i.e. when temporary files get deleted) but you're able to create 1 filesystem for temporary space and have it accessible (mounted) below both /tmp and /var/tmp. This means that they will share the same purging policy and will be easier to administer. The best benefit of all is that, because any user can occupy space below /tmp and /var/tmp, that the maximum space a user can consume is limited by the size of the filesystem and a full /tmp and /var/tmp won't compromise the integrity of your system or security.

- **Create a Logical Volume in size for /home big enough for your user home directories**

If your users won't be using their home directories, we still recommend creating a separate filesystem for /home, but naturally use a smaller size. Having normal users who are able to save data in their home directories which is part of the / filesystem poses a security and system integrity risk to your system.

- **Create a Logical Volume for /usr which is 10GiB in size**

The /usr filesystem is a 2nd tier in the Linux Filesystem Hierarchy Standard and contains items that influence the behavior of the system but these items are not needed at system boot time.

- **Create a Logical Volume for /var which is 10GiB in size**

/var contains all kinds of variable data like spools, logs, databases, caches and your internet services. Depending on the role of the server you may want to have additional filesystems for these but at a minimum you would want /var to be a separate filesystem.

- **Create a Logical Volume for /var/log which is an appropriate size for your environment**

Log files are stored below /var/log and depending on what is happening on your system, these files may get very big so account for this. If the /var/log filesystem is full at least you are able to provide space for other things like spools, databases, home directories etc. If /var/log is not a separate filesystem it poses a security and system integrity risk to your system.

- **Create a Logical Volume for the / filesystem which is 15GiB in size**

The root filesystem is a major filesystem and certain directories (like /bin, /sbin, /lib, /etc and /dev) must be part of it. The root filesystem however doesn’t contain much - it essentially has “everything else” as part of it.

Filesystem Labs

An engineer has recently devised the following filesystem layout for our organization’s mail server. How can a DOS attack be used to target the filesystem?

/
/home
/var/mail
swap

Server specific configurations

HTTP and FTP servers:

The HTTP has a default document root at **/var/www** and for FTP this is **/var/ftp** so if your server will fulfil the role of a dedicated HTTP and/or FTP server consider having separate Logical Volumes for these.

Proxy servers:

The key component of your proxy server is the cache directory, therefore you want to provide a separate filesystem for caching. Failing to do so could result in your cache directory consuming all the space on the filesystem which it is a part of or because other components of your system is so busy, you could have very little space left for caching. The Squid proxy server typically uses **/var/spool/squid** as its cache directory. Naturally we would create a Logical Volume for this filesystem and the size would be dependent on how long we would want an object to remain in cache for.

Internet speed	20mbps
TTL for objects in cache	5 days
Hours in a working day	9

20mbps translates to 2.5MBps and if your line is that busy for 9 hours per day that means that your cache could grow by 81GB per day or 405GB per week. So /var/spool/squid could be created as a 405GB Logical Volume.

Mail servers:

The spool directory for your mail server would be /var/spool/mail and this should be created as a separate Logical Volume. The size is dependent on the number of emails each user expects to receive per day. It is possible to allocate a quota for user mailboxes to limit usage for their email.

Database servers:

Because of the array of database daemons we have available to use, it’s hard to cover them all but typically databases are stored below /var/lib. We will cover the 2 most popular Open Source databases, namely MySQL and PostGreSQL. Naturally a Logical Volume would be used and the size would be dependent on the records you

expect in your database. MySQL would store the actual database in /var/lib/mysql and PostGreSQL would use /var/lib/postgresql
Additional considerations:

- **Create a Logical Volume for /opt which is about 1GiB in size**

Some 3rd party vendors specify that their software is installed below /opt. Should this not be the case, you may safely omit this guideline and have the /opt directory as part of the / filesystem.

Managing Networking

NetworkManager

While NetworkManager is fantastic for managing different types of networking on a desktop or laptop, it isn't well suited to enterprise level servers and as such we recommend disabling it.

While NetworkManager is NOT installed by default, it is implemented as a process rather than a daemon. To verify if NetworkManager is running use

```
# ps aux | grep NetworkManager
```

Traditional Networking

Once NetworkManager is disabled we can start working with the traditional networking service which has its configuration files stored below **/etc/sysconfig/network**

For each interface you want configured, you will need a configuration file called ***ifcfg-<interface name>***

If you have an interface called **eth0** then you would need a configuration file called **ifcfg-eth0** in order to have networking.

Inside **/etc/sysconfig/network/ifcfg-*interfacename:***

```
BOOTPROTO=
IPADDR=
NETMASK=
STARTMODE=
```

For example, to configure eth0 with a static IP address of 172.17.6.23 with a 16 bit subnet mask using DNS servers 8.8.8.8 and 8.8.4.4 with 172.17.0.1 as a gateway we would create a file called **/etc/sysconfig/network/ifcfg-eth0** with the following content:

```
BOOTPROTO=static
IPADDR=192.168.122.167
```

```
NETMASK=255.255.255.0
STARTMODE=auto
```

For DHCP configuration of an interface one would create a configuration file representing the interface with a minimum of 2 configuration parameters as follows:

```
BOOTPROTO=dhcp
STARTMODE=auto
```

To finalize the configuration and make it effective restart the network service:

```
# service network restart

or

# rcnetwork restart
```

To verify that your interfaces have a IP addresses we could use the command:

```
# ip address show
```

which could be abbreviated to:

```
# ip a s
```

which could even further be abbreviated to:

```
# ip a
```

To view the IP address of a specific interface like eth0:

```
# ip a s eth0
```

Should you need to take the interface down, use the command:

```
# ip link set eth0 down
```

And to bring it back up:

```
# ip link set eth0 up
```

Bridging

Note: An alternate configuration is to set up a bridge. This is especially useful with working with virtual machines and logical networks. The concept is easy enough. You are creating a layer of abstraction so that a physical device. So, the configuration of the physical device would look something like this:

```
DEVICE=eth0
ONBOOT=yes
ETHTOOL_OPTS="autoneg off speed 100 duplex full"
BRIDGE=br0
HWADDR=a1:b2:c3:d4:e5:f6
```

Then, the bridge configuration would look something like this:

```
DEVICE=br0
TYPE=Bridge
BOOTPROTO=none
ONBOOT=yes
IPADDR=192.168.0.5
NETMASK=255.255.255.0
DELAY=0
```

Notice that this is where the ip address and protocol are assigned (not in the config file of the physical device).

Routing

While your interface configuration file contains the default gateway, your system may have other routes configured. Create a file called **/etc/sysconfig/ifroute-eth0** if the route is applicable for eth0

The syntax is:

network gateway netmask interface

For example:

```
192.168.88.0    192.168.122.2    255.255.255.0 eth0
```

To finalize the configuration and make it effective restart the network service:

```
# service network restart

or

# rcnetwork restart
```

To verify that your routes have been set, use the command:

```
# ip route show
```

Which could be abbreviated to:

```
# ip r s
```

which could even further be abbreviated to:

```
# ip r
```

Setting your system's hostname

The name of your system is defined inside a configuration file called **/etc/HOSTNAME**

Using the command **hostname** does not persistently set your hostname and your system would have its name set back to what the above file specifies on reboot.

Networking tools

To test	We use	Example
LAN connectivity	ping	ping 192.168.0.1
WAN connectivity (routing)	traceroute	traceroute www.google.com
DNS	dig host nslookup	dig mx enterpriselinux.pro host www.redhat.com nslookup www.fedoraproject.org

Networking Labs

Open a terminal and download the following file using the command:

wget http://labs.enterpriselinux.pro/SUSE_networking.sh -O /usr/local/bin/networking.sh

!!! NOTE !!!

That is a capital O not the number zero.

Now in that same terminal run: networking.sh

Your task is now to setup networking on your machine as follows:

IP ADDRESS	192.168.0.x
SUBNET MASK	255.255.255.0
GATEWAY	192.168.122.1
DNS 1	192.168.0.254
DNS 2	8.8.8.8
DNS 3	8.8.4.4
HOSTNAME	yourname.example.com
ADDITIONAL ROUTE 1	10.0.0.0/255.255.255.0 via 192.168.0.253

ADDITIONAL ROUTE 2	172.17.0.0/255.255.0.0 via 192.168.0.252

When you are complete restart the network service to verify your work.

Determine the following using DNS lookups:

- who owns thelinuxclub.com?

Answer	

- the MX record(s) of nightmare.com

Answer	

- how many IP addresses does www.google.com point to?

Answer	

- What is the last router before www.tedmosbyisajerk.com?

Answer	

Process Management

ps

This is the most basic command for getting information about running services.

```
# ps -ef | grep <keyword>
```

jobs

Jobs are processes that are confined to specific bash instances, or terminals.

jobs is a command that tells you which jobs are running in a terminal.

kill

If a process becomes problematic, it may become necessary to kill that process and start a new instance. This is far more preferred to having the system lock up or having to reboot (as is the case with certain operating systems).

To kill a process: **kill -9 <pid>**

To kill a job: **kill %<job number>**

fg

This command will bring a command that is running in the background to the foreground.

fg %<job number>

bg

Processes that run in the background allow the terminal that they are running in to be used for other things while the background process is running. Processes can be run in the background by adding an ampersand (&) after the command. In order to change a job from the foreground to the background you must first interrupt the job and with the key combo Control+z.

time

The time command is a good way to find out how long it takes for a specified command, or instance of a process, to run. This command tells you the total time for the process to run, the time it took to run in the kernel space and the time it took in the user space. This can be very useful information when you are doing performance tuning on a system to benchmark your success.

pstree

This command shows the relationship of running processes to each other. For example, the visual output will tell you what a parent of child process of other processes exist.

top

The top command will show you how much of the various resources are being exhausted on specific processes. The top command output is very similar to the System Monitor graphical tool. It can also be sorted to show the order of the most resource-intense processes. The columns can be selected

Process Management Labs

Open a terminal and execute the following command: **sleep 30**

Now attempt to run **ls -l** and then try to execute **cat /etc/*release**

Did the commands **ls -l** and cat **/etc/*release** execute immediately?

Answer	

Now repeat the steps above but this time add an **&** after the **sleep** command. Did the commands **ls -l** and cat **/etc/*release** execute immediately?

Answer	

What is the significance of the **&**?

Answer	

We’re getting somewhere now, so let’s try the following. run the following command:
sleep 100 and do not use an **&** but this time hit **CTRL + z**. What happened?

Answer	

Is the **sleep 100** command running?

Answer	

!!! TIP !!!

If you’re not sure use the the command **pidof sleep**

So what you can determine now is that the & detaches the command from your shell (in other words it’s still running). Many think that the & puts a process into the background. A background process is a process which is not being serviced by the CPU. In other words it has a PID, has been allocated memory resources but it isn’t getting CPU cycles.

Now take the process out of the background using commands which you’ve just learned.

!!! NOTE !!!

We like using the kill command to put processes in the background and return them to the foreground.

To send a process to the background: kill -19 <process id>
To bring a process to the foreground: kill -18 <process id>

Monitoring Resources

Four is the magic number when it comes to resources. There are **four** major resources. There are **four** places where the settings of these resources are persistently set. After reviewing this section you will likely be so disappointed with the default configuration of Linux that you want to take **four** Advil (TM) and before you actually do anything to changes the settings of your system you want to benchmark what the current performance of the resources are doing... be**fore** you do anything.

This section is all about getting information from the system. There are many utilities to help us with that. Here are a few of them for the different resources.

CPU

The Central Processing Unit, chip, processor, etc. is the workhorse of the system. The unit of measure we use for the CPU is Hertz (Hz, or cycles per second). Ever cycle of the cpu carries an instruction to be accomplish. Therefore, we ought to be most concerned with the efficiency of cycles on the CPU. A 3 GHz processor is not uncommon today. Many times a system will be equipped with two or four of these processors.

Real-time output from the kernel gives us a lot of information about our CPU.

```
# cat /proc/cpuinfo
```

The above command the solicits cpu information from the kernel.

```
# ps axo comm,pid,%cpu
```

The above command wiill also show how much of the processor’s resources each processes is using.

Memory

There are different kinds of memory. Physical RAM is what most people refer to as memory. Swap space is hard drive storage that has been dedicated to the overflow memory. Virtual memory has theoretical limits and maps filesystem usage to other memory. The philosophies that seems to work best for most administrators are 1) the more RAM, the better. 2) to have between 8 and 12 Gigabytes of swap space, but try not to use any of it. 3) the default settings are for stability, not performance. Some commands that can be used to monitor memory are:

```
# cat /proc/meminfo
```

```
# ps axo comm,pid,%mem
```

Disk

Hard drive storage is something that evolves fairly quickly. With the advent of cloud computing, many end users are becoming less aware of where their files are physically located. Because people can be online anywhere, netbooks (with little or no storage) are being adopted.

In enterprise environments there are a lot of companies that have adopted the concept of a cloud storage and have implemented their own private cloud. In such environments, administrators still must be very involved in managing storage to support the could.

On Linux, to quickly see how much storage devices have and the capacity of those devices the command **df -h** will give this information at a glance. Here is what the output looks like:

```
# df -h
Filesystem                      Size  Used Avail Use% Mounted on
/dev/mapper/vg_name-storage      50G   35G   13G  74%    /storage
```

Network

Although there is a lot of third-party software to monitor network traffic, Linux has several good utilities to show us what our network is doing.

While netstat, mtr, tcpdump and iptables will tell what a system’s network traffic is doing, there are a few commands that show us how well the network is handling that traffic.

iostat -n gives network statistics. The kinds of things to observe and compare are the reads and writes. **sar** reports give network statistics, but for really good usable information we recommend **Nagios**.

Monitoring Resources Labs

Determine the sizes of the following directories:

Directory	Size

/usr/share/doc	
/etc/sysconfig/selinux	
/proc	

Determine the following about your system using commands:

Type of display adapter	
Speed of your CPU	
MAC address of your network card	
Amount of physical RAM	

What is the first process that was started on your system?

Answer	

Software Management on SUSE Linux

SUSE uses a framework which is very similar to **yum** called **zypper** which is also RPM based.

```
# zypper install -y nmap
```

This tells zypper to **install** the package **nmap** without prompting you to confirm **(-y)** the transaction.

But where does it get the software from?

zypper conducts all of its transactions against the repositories defined in **/etc/zypp/repos.d**

A repository is a collection of software packages which may exist in the form of:

- a directory on your computer
- a directory on another computer which is shared for you using NFS or SAMBA
- a http(s) site
- a (s)ftp site

If your system is not connected to a repository then you will not be able to install software unless you've downloaded the package file manually.

So how do I find out what software is available for me to install? Well that answer could be very big but if you really want to try, use the command:

```
# zypper packages
```

If you want to see which software packages have already been installed on your system, use the command:

```
# zypper packages -i
```

If you want to search for packages whose name or summary contains the term “audit” run the following:

```
# zypper search audit

Loading repository data...
Reading installed packages...

S | Name                | Summary                                                | Type
--+---------------------+--------------------------------------------------+-----------
i | audit               | User Space Tools for Kernel Auditing        | package
  | audit                                 | User Space Tools for Kernel Auditing             |
srcpackage
  | audit-audispd-plugins | Default plugins for the audit dispatcher         | package
i | audit-libs          | User Space Tools for Kernel Auditing        | package
i | audit-libs-32bit    | User Space Tools for Kernel Auditing        | package
  | audit-libs-python   | Python Bindings for libaudit                          | package
  | audit-secondary     | Python Bindings for libaudit                          | srcpackage
  | pwdutils-plugin-audit | Plugin for the Linux Audit-Subsystem            | package
i | yast2-audit-laf     | YaST2 - Configuration of Linux Auditing (LAF)     | package
  | yast2-audit-laf     | YaST2 - Configuration of Linux Auditing (LAF)     | srcpackage
```

To get detailed information about a package, then use:

```
# zypper info audit

Loading repository data...
Reading installed packages...


Information for package audit:

Repository: sles11.3
Name: audit
Version: 1.8-0.30.1
Arch: x86_64
Vendor: SUSE LINUX Products GmbH, Nuernberg, Germany
```


```
Support Level: Level 3
Installed: Yes
Status: up-to-date
Installed Size: 915.0 KiB
Summary: User Space Tools for Kernel Auditing
Description:
The audit package contains the user space utilities for storing and
processing the audit records generated by the audit subsystem in the
Linux kernel.
Authors:
--------
        Steve Grubb <sgrubb@redhat.com>
```

Removing packages are really easy, as easy as:

```
# zypper remove nmap

Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following package is going to be REMOVED:
  nmap

1 package to remove.
After the operation, 4.6 MiB will be freed.
Continue? [y/n/?] (y): y
Removing nmap-4.75-1.28.1 [done]
```

!!! NOTE !!!

Here we get prompted as to continue the transaction and we do NOT want to answer yes automatically. Dependencies are “needed software packages” and when you install a package using zypper, then the dependencies are automatically installed as they are needed by the software which you want to install. Removing a dependency, results in removing all the software packages

which require that dependency. Do you really want to automate this?

Updating packages is also possible with the command zypper.

```
# zypper update
Loading repository data...
Reading installed packages...

Nothing to do.
```

If you want to update your entire system (be careful with this one and always do your testing before going live):

```
# zypper update -y
```

Packages may also be managed by group! A package group is a grouping of individual purpose which serve a common purpose.

To see all the available package groups use the command:

```
# zypper patterns
Loading repository data...
Reading installed packages...
S | Name              | Version      | Repository | Dependency
--+-------------------+------------------ ---+-------------------+-----------
  | 32bit       | 11-38.44.33    | sles11.3   |
  | Basis-Devel       | 11-38.44.33 | sles11.3   |
  | Dom0              | 11-38.44.33 | sles11.3   |
  | Dom0_KVM | 11-38.44.33 | sles11.3   |
i | Minimal           | 11-38.44.33 | sles11.3   |
i | WBEM              | 11-38.44.33 | sles11.3   |
i | apparmor     | 11-38.44.33 | sles11.3   |
i | base        | 11-38.44.33 | sles11.3   |
  | dhcp_dns_server  | 11-38.44.33 | sles11.3   |
  | directory_server | 11-38.44.33 | sles11.3   |
i | documentation   | 11-38.44.33 | sles11.3   |
  | file_server       | 11-38.44.33 | sles11.3   |
```

```
  | gateway_server   | 11-38.44.33 | sles11.3   |
i | gnome            | 11-38.44.33 | sles11.3   |
  | kde         | 11-38.44.33 | sles11.3   |
  | kvm_server | 11-38.44.33 | sles11.3   |
  | lamp_server| 11-38.44.33 | sles11.3   |
  | mail_server       | 11-38.44.33 | sles11.3   |
  | ofed        | 11-38.44.33 | sles11.3   |
  | oracle_server     | 11-38.44.33 | sles11.3   |
i | print_server      | 11-38.44.33 | sles11.3   |
  | sap_server        | 11-38.44.33 | sles11.3   |
i | x11         | 11-38.44.33 | sles11.3   |
  | xen_server   | 11-38.44.33 | sles11.3   |
```

We've spotted a group with a short name **mail_server** in the list and will work with that package group.

If you want to know more about a package group then you can query its metadata:

```
# zypper info -t pattern mail_server

Loading repository data...
Reading installed packages...

Information for pattern mail_server:

Repository: sles11.3
Name: mail_server
Version: 11-38.44.33
Arch: x86_64
Vendor: SUSE LINUX Products GmbH, Nuernberg, Germany
Installed: No
Summary: Mail and News Server
Description:
Software to set up electronic mail and message services to handle e-mail, mailing, and
news lists, including a virus scanner to scan messages at the server level.
Contents:

S | Name       | Type  | Dependency
--+-------------+---------+-----------
```

```
| mailman     | package |
| spamassassin | package |
| clamav      | package |
| inn         | package |
| cyrus-imapd    | package |
| amavisd-new  | package |
| vacation    | package |
```

To install a package group:

```
# zypper install -t pattern -y mail_server
```

Similar commands apply when you want to update or remove package groups.

All zypper commands are conducted against a repository and these definitions are stored in the directory **/etc/zypp/repos.d**

The rule is that your repository definitions are stored in files ending in **.repo** in that directory.

Examples:
/etc/zypp/repos.d/foo.repo
/etc/zypp/repos.d/bar.repo

Multiple repository definitions may be grouped in 1 file and each repository definition consists of a minimum of 4 lines:

```
[shortname]
name=this is a long name which may be more descriptive
baseurl=URL of repository
```

If you want to verify that software from this repository has been only modified by its maintainer then you need to change to:

```
gpgcheck=1
gpgkey=URL of public key
```

Example:

```
# cat /etc/zypp/repos.d/foo.repo
[foo]
name=foo repository used in testing
baseurl=ftp://192.168.122.1/pub/sles11.3
gpgcheck=1
gpgkey=ftp://192.168.122.1/pub/sles11.3/gpg-pubkey-307e3d54-4be01a65.asc
```

Software Management on SUSE Linux Labs

Using only the **zypper** command, install the following:

The package **nmap**
The package group **TeX support**
The package which provides the configuration file **/etc/my.cnf**
The package which provides you with the executable **semanage**

Now using the **zypper** command, remove the package **vnc**

What is the purpose of the package **xeyes**?

Answer	

Service Management

Installing software in most cases is not very useful unless you know how to manage the software that was just installed. This does not apply, of course, to packages that only contain documentation, like kernel-doc. This section discusses how start and stop services, determine which services are running and listening on specific ports and how to ensure that services start (or not) when the system boots.

service

The **service** command's function varies between services. All services will at least allow you to start and stop with the service command. The service command **service network restart** could also be expressed as **/etc/init.d/network restart**. They do the exact same thing.

The useful element of teaching the **service** command is that it can be used on Enterprise Linux, SUSE and Ubuntu based distributions.

chkconfig

Pronounced as “check config,” this command will persistently set whether a service starts or not in the different run levels. Under the hood, chkconfig changes a symbolic link in the runlevel specific directories which signifies to start or kill the service as the operating system enters that run level. This command can be run generically to apply to many runlevels at once, or for one specific runlevel.

```
# chkconfig postfix on
```

The above command applies to all of the runlevels specified in the actual “init script” for that service.

```
# chkconfig --level 3 postfix off
```

The previous command would only turn postfix off as the system enters into runlevel 3.

netstat

The netstat command can tell you many things. One of them is which services are listening on which ports.

netstat -plunt is a common command for finding which ports are being used.

netstat -plunt | grep -i listen will show only the services that are listening on specific ports. This is useful information for protecting network ports through a firewall. With this information you will know which ports need rules to protect them. In other words, which ports are exposed to the outside.

!!! TIP !!!
Sometimes it’s useful to take the options of a command and try to formulate a word out of it, afterall, in most cases the order of the options aren’t important. So **plunt** isn’t a word it’s a combination of the options we use.

/etc/services

This file gives a generic listing of ports used by different services.

```
# rpcinfo -p
```

Like netstat, this command tells you which ports are being accessed, but specifically for remote procedure call (rpc) services like nfs.

Service Management Labs

Record the amount of free memory on your server:

Answer	

Now let’s stop some services which we don’t really need right now:

- avahi-daemon
- cgconfig
- fcoe
- firstboot
- iptables
- iscsi

- iscsid
- spice-vdagentd
- rpcidmapd
- rpcgssd
- rpcbind

How much free memory do you have now?

Answer	

Will these services which you've just stopped be started again when the system next boots?

Answer	

What do you need to do to change this?

Answer	

autofs

The autofs framework is used to make directories available (mount) on an as-needed basis which saves bandwidth and memory space. When the mounts are not used for some time - the default is 5 minutes - then it is unmounted.

autofs is create for provisioning home directories via NFS or SAMBA to users who authenticate via LDAP whose directories reside on shared storage.

Given the scenario below, we will use **autofs** to provision directories which our users need.

All our users make use of an NFS share /data/public on a server ares.enterpriselinux.pro. Provisioning the share via /etc/fstab could result in errors when the computer boots up as the user may not be on the same network as ares.enterpriselinux.pro. Only when the users attempt to access ~/public should the autofs service on their local computer attempt to mount the remote share.

Installation

Should autofs not be installed use YUM to install it from the standard SUSE Linux repository:

```
# zypper install -y autofs
```

Now activate it to start persistently:

```
# chkconfig autofs on
```

Configuration

autofs uses a control file called /etc/auto.master which typically defines the use of 2 columns:

Parent directory (absolute path)	Control file

The parent directory defines where you would go to access the child (the child itself is defined in the control file). The control file contains the definitions on what is to be mounted and from where.

Example:

~	/etc/auto.public

When something (a child) is accessed below the parent ~, the /etc/auto.public file defines what is to be mounted.

The configuration isn't complete at this stage - we still need to create **/etc/auto.public**

The control file typically uses 3 columns for its definition:

Child (relative to the parent director)	Options	Source

The child specifies where your mount is accessible from, the options specify how the mount is made available and the source is what you're mounting.

Example:

public	-ro	ares.enterpriselinux.pro:/data/public

Once these configurations have been made, simple restart or reload the autofs service and change directory to ~/public. The **mount | grep public** command should indicate that your autofs mount was successful.

---

autofs supports the use of wildcards too and this is particularly useful when having to mount user home directories so let's have a look at an example in context.

Enterprise Linux Professionals employs 78 administration and call center staff who typically use a thin client instead of a traditional computer. They authenticate to a centralized authentication server (like LDAP) and their home directories are provided by an NFS server ares.enterpriselinux.pro at the share /export/rhome/$USERNAME
Our configuration:

First we need to determine where the home directory for a LDAP user is expected:

```
# getent passwd fred
```

fred:*:50000:50000:Fred Flintstone:/home/fred:/bin/bash

As you can see, fred's home directory is expected at /home/fred but the actual directory is /export/rhome/fred on a server called ares.enterpriselinux.pro

In /etc/auto.master:

/home	/etc/auto.home

This entry specifies that when a child is accessed below the parent **/home**, the control file **/etc/auto.home** is called to determine what is mounted and where.

In /etc/auto.home

*	-rw	ares.enterpriselinux.pro:/export/rhome/&

We can't refer to the child by its actual name (fred) because if we had to do so we'd need 77 other similar entries for all our current users. This also doesn't cater for users who are added in future. Instead, in column 1, we use a wildcard which represents a directory below the parent **/home.**

We then do a read/write mount of **ares.enterpriselinux.pro** which has a share called **/export/rhome/&**

The & in column 3 represents whatever was asked for in column 2. So because we asked for the directory **fred** (which is expected to be below /home) we will do a read/write mount of **ares.enterpriselinux.pro**'s share called **/export/rhome/fred**

The use of wildcards here extended functionality because we don't need multiple entries for each user and it's scalable when we add additional future users.

Filtering traffic using the firewall

yast firewall for SUSE Linux

yast firewall (ncurses version) or yast2 firewall (GTK version) is a graphically oriented tool aimed at setting up your firewall. It is simple enough for you to use immediately at this stage of your training but we will discuss this in more detail in ELSA II.

Secure Communication

SSH

SSH is used as it is more secure than telnet as it makes use of encryption. Telnet transmits all data in clear text which means that a MITM (man in the middle) can intercept data including your credentials.

A few basics about SSH:

- it is an acronym for secure shell
- it uses 22/tcp
- it uses symmetric cryptography by default
- data is encrypted using session keys

When a SSH client requests a new connection to the SSH server for the first time, the user is prompted to trust that the SSH server is in fact the real SSH server the client intended communicating with. This is done by the SSH server sending its digital fingerprint to the client and the user can choose whether to accept it (trust it) or not.

A mapping between the SSH server's name, IP address and digital fingerprint is stored on the client at **~/.ssh/known_hosts**

Should a discrepancy exist, the client is warned that a possible MITM could be at play. Someone malicious could have initiated a Denial of Service attack against the SSH server thereby making its IP address unavailable on the network. The attacker would then configure their SSH server with that IP address and listen for connections. While the attacker's SSH server cannot authenticate you, it does prompt you for your password which you would then willingly give should the verification mechanism exist. Thankfully, we're protected against such attacks thanks to **~/.ssh/known_hosts**

Having said that, should you get a warning message about a possible MITM attack, remember that if you reinstall the server without restoring the original identity, the client will suspect a MITM even though this is not the case. When preparing to reinstall a server, backup all the files in **/etc/ssh/*key*** and restore them when the new server is up.

Establishing an SSH session

The basic syntax is:

```
$ ssh username@hostname
```

Should the username be omitted, then the username of the currently logged on user is used to authenticate to the SSH server.

To forward X Windows data from the SSH server to your local client, use the **-X** option with the ssh command:

```
$ ssh -X username@hostname
```

Any X Windows programs started via the SSH session will use the resources of the remote computer but will display on your computer.

How it works with telnet:

1. User initiates connection with the telnet server.
2. Telnet server responds requesting the credentials.

What could go wrong?

How do you know that the telnet server which you're connecting to really is the one you intend connecting to? Another malicious user could initiate a DoS attack against the real telnet server - thereby making its IP address offline - and then the attacker assigns that IP address to his own computer.

Even without the above scenario, telnet is a plaintext protocol so all data is transmitted in clear text over the network. Using a program like **ettercap** we could intercept the data between the telnet client and the telnet server thereby seeing the username and password being transmitted.

How it works with ssh with only symmetric cryptography (the default behavior of the SSH server):

1. User initiates connection with the ssh server.

2. If it is a new connection, the client is prompted to trust that the SSH server is indeed the one which the user expects connecting to.This is done by the SSH server submitting its digital fingerprint to the client and should the client trust the server, the mapping between the IP address and digital fingerprint is saved on the client in **~/.ssh/known_hosts**.

3. If the connection was previously trusted, the client verifies that the digital fingerprint transmitted by the SSH server matches the one previously saved in **~/.ssh/known_hosts**. Should the verification fail, the connection is terminated warning of a possible MITM attack.

4. Given that all is good, the SSH server generates a symmetric cryptographic key (a key that both encrypts and decrypts data) which is transmitted to the user. The key is uniquely generated per session so should the client disconnect, then that key will be invalid. The common name for this key is a **session key**.

5. The client then encrypts their credentials with this key and submits the encrypted data across the network to the ssh server.

6. The SSH server then uses the same session key to decrypt data and verify the authenticity of the client.

7. All data is subsequently encrypted and decrypted in this way.

What could go wrong?

Because SSH by default uses host key verification, it is not possible to spoof the SSH server’s IP address as the digital fingerprint cannot be replicated without server’s

private key. But there's a more fundamental flaw in the default behaviour of your SSH server: the session key is transmitted in plaintext!

A MITM could intercept the transmission of the session key when the SSH client initiates a connection to the SSH server, then when the SSH client authenticates, the MITM could intercept that encrypted data and decrypt it as he has a copy of the session key.

Why you should configure the SSH server to only work with Symmetric and Asymmetric cryptographic keys:

1. User initiates connection with the ssh server.

2. If it is a new connection, the client is prompted to trust that the SSH server is indeed the one which the user expects connecting to.This is done by the SSH server submitting its digital fingerprint to the client and should the client trust the server, the mapping between the IP address and digital fingerprint is saved on the client in **~/.ssh/known_hosts**.

3. If the connection was previously trusted, the client verifies that the digital fingerprint transmitted by the SSH server matches the one previously saved in **~/.ssh/known_hosts**. Should the verification fail, the connection is terminated warning of a possible MITM attack.

4. Should all be fine with the authenticity of the SSH server, the SSH server checks to see if a public key exists for user who wishes to authenticate inside **~/.ssh/authorized_keys** on the SSH server. If a public key exists then the SSH server generates a session key and before transmitting the session key to the user, the SSH server encrypts it with the user's public key. If no public key exists for the user (in ~/.ssh/authorized_keys) then the user is prompted for their password and session keys are used as explained above. If PasswordAuthentication is set to no and the user does not have a public key in ~/.ssh/authorized keys then the user cannot login.

5. The SSH client receives the session key in an encrypted form and the user then has to use the partnering private key in order to decrypt the data. The use of a private key is subject to quoting the optional passphrase. Once the

SSH client has decrypted the session key, all data is symmetrically encrypted using the session keys on both the SSH server and client.

Why this is safer:

Previously a MITM was able to intercept the session key being transmitted between the SSH server and SSH client as the session key was transmitted in clear text. Now that the session key is first encrypted with the user's public key, only the user who possesses the accompanying private key can decrypt it and use the session key.

Configuring SSH to use Public/Private key authentication

On the client:

```
$ ssh-keygen
```

This will prompt you to create a public/private key pair. Should one not specify the type and size, a 2048 bit RSA key pair will be generated called ~/.ssh/id_rsa.pub and ~/.ssh/id_rsa for the public and private keys.

!!! NOTE !!!
If no input is provided to the questions which the ssh-keygen command prompts of you, then the default values will be used. These defaults appear alongside the prompt in brackets.

!!! TIP 1 !!!
Use a passphrase to protect the use of your private key.

!!! TIP 2 !!!

To make your SSH key implementation more secure, stay away from the default names and consider storing your private key as something arbitrary, for example:

~/Documents/Personal/Recipes/Desserts/Apple_Pie.doc

Now that the keys have been created, we need to install the public key on the SSH server.

```
$ ssh-copy-id <ip address of server>
```

!!! NOTE !!!
If you gave your keys different names then you will need to specify the name of your public key using the **-i** option with the **ssh-copy-id** command

```
$ ssh-copy-id ricardo@apollo
Warning: Permanently added '192.168.122.100' (RSA) to the list of known hosts.
ricardo@apollo's password:
Now try logging into the machine, with "ssh 'ricardo@apollo'", and check in:

 ~/.ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
```

Configuring the SSH daemon

The behavior of the SSH daemon is defined in **/etc/ssh/sshd_config** and is only editable by the user root.

Whenever the **ssh** (client) command is used, the global configuration file **/etc/ssh/ssh_config** (editable only by root) is applied and then normal users can tailor how their ssh command behaves by creating a file called **~/.ssh/config**

Basic security precautions and hardening SSHd

!!! NOTE All configurations are to be implemented in /etc/ssh/sshd_config !!!

Consider running the SSH daemon on a different port:
Port 2222

We never want anyone to log in directly as root:
PermitRootLogin no

Consider limiting the users who are allowed to login to the server via SSH:

AllowGroups sshusers

While session keys are used by default, they are not considered secure so allow only authentication using a Public/Private key combination:

PasswordAuthentication no

SSH allows administrators to set an idle timeout interval. After this interval has passed, the idle user will be logged out:
ClientAliveInterval 15
ClientAliveCountMax 3

Given the above, a client would be disconnected after an idle time of 45 seconds. This will work when SSH clients do not send “keepalives”.

If the client is sending keepalives then add the following to **/etc/bashrc**

TMOUT=600

Any user who is idle for 10 minutes will be automatically disconnected regardless of how they are logged into the system.

Configure the SSHd to use its own SysLog facility to make logging easier:

SyslogFacility local0

Use **fail2ban** (EPEL or http://fail2ban.sourceforge.net) to ban IP addresses (temporarily) who repeatedly fail to authenticate. This is extremely useful to protect yourself against brute force attacks.

Limit the number of connection attempts to your SSHd’s port:

```
# iptables -I INPUT -p tcp --dport 22 -i eth0 -m state --state NEW -m recent  --set

# iptables -I INPUT -p tcp --dport 22 -i eth0 -m state --state NEW -m recent
--update --seconds 60 --hitcount 5 -j DROP

# service iptables save
```


Setup SSH for port knocking using iptables. What this means is that your SSH server will not accept connections on its port (in our case port 22) unless you have sent TCP packets to another port first (in our case port 445).

First you need to ensure that you have the **xt_recent** kernel module loaded:

```
# lsmod | grep xt_recent
```

If the module isn't loaded, then load it and set it to load persistently - this is covered in ELSA II.

In **/etc/sysconfig/iptables**, take inspiration from the following:

```
-A INPUT -p icmp --icmp-type any -j ACCEPT
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -m recent --rcheck
--seconds 5 --name SSH -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 389 -m recent --name SSH
--remove -j DROP
-A INPUT -m state --state NEW -m tcp -p tcp --dport 445 -m recent --name SSH
--set -j DROP
-A INPUT -m state --state NEW -m tcp -p tcp --dport 636 -m recent --name SSH
--remove -j DROP
-A INPUT -j DROP
```

This line allows connectivity to tcp/22 for 5 seconds once a connection type called SSH exists.

Should a connection type called SSH exist, this line removes it when a connection is made to tcp/389.

Once a connection is made to tcp/445, a connection type called SSH is created.

Should a connection type called SSH exist, this line removes it when a connection is made to tcp/389

telnet

In the past, we used the telnet command to connect to a telnet server in order to logon to a remote system and issue commands to it. Because telnet is a plaintext protocol, we no longer use it for remote shells but we certainly do use the telnet client command to make connections to remote systems on other ports.

Let's make a connection to the SMTP server on 192.168.122.100 and send an email root using the standard SMTP commands.

```
$ telnet 192.168.122.100 25
Trying 192.168.122.100...
Connected to 192.168.122.100.
Escape character is '^]'.
220 virtual1.example.com ESMTP Postfix
mail from: fred@flintstones.org
```

```
250 2.1.0 Ok
rcpt to: root
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
Yaba daba doo!
.
250 2.0.0 Ok: queued as CE0DD46A6
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

(The period on a blank line is used to end the data submission and send the email.)

Secure Communications Labs

Why should telnet not be used?

Answer	

True or false: By default SSH is secured out of the box and MITM attacks are not possible.

Answer	

What is the name of a newly created RSA public key?

Answer	

What is the name of a newly created DSA private key?

Answer	

When you create a public/private key pair - where are the files stored?

Answer	

How many public keys can be stored in the authorized_keys file?

Answer	

Create a set of RSA based SSH keys for your user account from your Physical Machine. Copy only the public key to your Virtual Machine. Make sure that your user account is a member of the group **wheel** on your VIrtual Machine . Do not use a passphrase for your private key.

Configure your Virtual Machine so that you no longer allow password authentication and only allow authentication for those who are members of the group **wheel** who have a valid public and private key pair.

True or False: It is possible to bind the SSH daemon to 23/tcp.

Answer	

What is the purpose of fail2ban?

Answer	

Virtualization

KVM

KVM is a virtualization infrastructure for the Linux Kernel. If you're using a the standard Linux kernel from version 2.6.20 then you have support for KVM.

To determine your Linux kernel version to meet the software requirements:

```
# uname -r
2.6.32-279.el6.x86_64
```

!!! NOTE !!!

KVM doesn't perform any emulation but allows the /dev/kvm interface to be used by a user-space program like QEMU to perform the emulation of hardware.

KVM does have 2 very specific hardware requirements: 64 bit CPU and support for hardware assisted virtualization which is enabled in the BIOS.

To verify that your system meets the hardware prerequisites for KVM:

```
# grep flags /proc/cpuinfo | uniq | grep -E --color 'vmx|lm|svm'
flags     : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp lm constant_tsc
arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni
pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2
x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt
pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase smep ermsf
```

Matching to the flag **lm** means that you have a **64 bit processor**, **vmx** means that **hardware assisted virtualization enabled in the BIOS** for an Intel processor and **svm** means the same for an AMD processor.

Now that we've met the software and hardware requirements, we're able to use QEMU to create and manage virtual machines.

QEMU

QEMU is your Quick EMUlator and emulates a hardware environment for user-level processes. Virtualization across architectures is not supported, so one cannot run an AMD guest on an Intel host.

Paravirtualization vs Full Virtualization

Full virtualization uses a special kind of software called a hypervisor. The hypervisor interacts directly with the physical server's CPU and disk space. It serves as a platform for the virtual servers' operating systems. The hypervisor keeps each virtual server completely independent and unaware of the other virtual servers running on the physical machine. Each guest server runs on its own OS - you can even have one guest running on Linux and another on Windows.
The hypervisor monitors the physical server's resources. As virtual servers run applications, the hypervisor relays resources from the physical machine to the appropriate virtual server. Hypervisors have their own processing needs, which means that the physical server must reserve some processing power and resources to run the hypervisor application. This can impact overall server performance and slow down applications.

The para-virtualization approach is a little different. Unlike the full virtualization technique, the guest servers in a para-virtualization system are aware of one another. A para-virtualization hypervisor doesn't need as much processing power to manage the guest operating systems, because each OS is already aware of the demands the other operating systems are placing on the physical server. The entire system works together as a cohesive unit.

Creating a Virtual Machine

A number of resources are available for creating Virtual Machines but we will focus on using the Virtual Machine Manager program called **virt-manager** and **virt-install.**

virt-manager is a GUI based program which allows you to easily create and further customize a virtual machine.

virt-install is a CLI based program which allows you to easily create a new virtual machine.

The command below creates a virtual machine called **dev.web** which uses **512M RAM** and **1 Virtual CPU** to install an Operating System from the URL **ftp://192.168.122.1/pub/rhel6.3** using the disk **/var/lib/images/dev.web.img**

```
# virt-install -n dev.web -r 512 --vcpus=1 -l
ftp://192.168.122.1/pub/rhel6.3 --disk /var/lib/images/dev.web.img
```

!!! TIP !!!
The disk used above was created as a sparse image file using the dd command:

```
# dd of=/var/lib/images/dev.web.img bs=1 seek=10G count=0
```

Connecting to a virtual machine's console

The easiest way to get a console for your virtual machine is to use **virt-manager** and then double-click the virtual machine that you want to open a console to.

However, most enterprise level systems run headless and you will need to know how to access a console for a virtual machine across a remote connection. For this we use the **virsh** command.

To see the virtual machines you're running:

```
# virsh list
Id      Name                    State
----------------------------------------------------
2       RHWS300                 running
```

To connect to a virtual machine you can either specify the Name (RHWS300) or the ID (2):

```
# virsh console 2
Connected to domain RHWS300
Escape character is ^]
```

Once you are done working on that virtual machine, don't forget to **logout** or **exit** otherwise you will remain logged in. To finally disconnect from the virtual machine's console send the keys **CTRL]** to it.

Should you type in your username and not get any response then it you don't have a serial console configured on that virtual machine. You will now need to use **virt-manager** to connect to the virtual machine, log on and open up **/boot/grub/grub.conf** and add the following at the end of the line which begins with the word **kernel**:

```
console=ttyS0
```

For example:

```
# cat /boot/grub/grub.conf
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#         all kernel and initrd paths are relative to /boot/, eg.
#         root (hd0,0)
#         kernel /vmlinuz-version ro root=/dev/mapper/VolGroup-lv_root
#         initrd /initrd-[generic-]version.img
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title SUSE Linux (2.6.32-279.el6.x86_64)
   root (hd0,0)
   kernel /vmlinuz-2.6.32-279.el6.x86_64 ro root=/dev/mapper/VolGroup-lv_root
rd_NO_LUKS LANG=en_US.UTF-8 rd_NO_MD rd_LVM_LV=VolGroup/lv_swap
SYSFONT=latarcyrheb-sun16 crashkernel=auto rd_LVM_LV=VolGroup/lv_root
KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet console=ttyS0
   initrd /initramfs-2.6.32-279.el6.x86_64.img
```

You will need to reboot in order to make this change effective.

Starting Virtual Machines

The default auto start behavior is that any virtual machine that was started when the physical machine (the host) was shutdown, will automatically start when the physical machine boots up.

This may not be what you want and if you only want virtual machines to start automatically if you explicitly tell it to, then you need to change the following.

In /etc/sysconfig/libvirt-guests change the line which says:

```
ON_BOOT=start
```

to

```
ON_BOOT=ignore
```

Then restart the libvirtd with the command:

```
# service libvirtd restart
```

Now we need to go to the virtual machines and instruct them to start automatically when the physical machine starts (regardless of the virtual machine's state when the physical machine was shutdown or rebooted). You can use **virt-manager** for this purpose or **virsh**.

Using **virt-manager**:

Select by double clicking the virtual machine you want to auto start, then click **VIEW > DETAILS > BOOT OPTIONS** and tick **Start virtual machine on host boot up**

Using **virsh**:

```
# virsh autostart NAME
```

or

```
# virsh autostart ID
```

To disable the autostart status of a virtual machine using **virsh** use:

```
# virsh autostart NAME or ID --disable
```

Virtualization Labs

What are the hardware requirements for running virtualization using KVM?

Answer	

Is it possible to run a Windows virtual machine on a Linux system?

Answer	

True or false: The **nx** flag is required for KVM virtualization.

Answer	

Use your favorite tool to create a new virtual machine which has a 5GB disk, 1 CPU and 512MB RAM using any version of an enterprise Linux distribution. To conserve space do a minimal installation.

When you are done make sure that you can make a console connection to that server using the command: **virsh console <name of virtual machine>** from your physical machine.

Being able to make a serial console connection to a virtual machine is important should you lose TCP/IP connectivity it.

Make sure that your virtual machine is automatically started regardless of whether it was running when the physical machine was rebooted.

Logging

Event notification is a critical subsystem of any Operating System environment and these events are stored as logs using a logging daemon. In SUSE Linux we use the **syslog-ng** for logging our events. This daemon has support for logging local events as well as events happening remotely on other systems on a TCP/IP network. Logs do not have to be stored as files and for larger deployments we recommend and support logging events to a databases such as MySQL or PostGreSQL.

How the daemon works is that it listens for a events being sent on your system and it is matched by 2 items - a facility and a priority. A facility represents a subsystem which is responsible for generating that message. These subsystems are predefined and the developer of a subsystem specifies whether 1) the subsystem supports the use of syslog, 2) which facility it uses by default and 3) whether the facility may be changed or if it’s statically linked.

Facilities

The facilities are predefined and are used as follows:

Facility	Subsystem it represents
auth	Activity related to requesting name and password (getty, su, login)

authpriv	Same as auth but logged to a file that can only be read by selected users
console	Used to capture messages that would generally be directed to the system console
cron	Messages from the cron task scheduler
daemon	System daemons
ftp	Messages relating to the ftp daemon
kern	Kernel messages
local0 to local7	Local facilities defined per site
lpr	Messages from the line printing system
mail	Messages relating to the mail system
news	Messages relating to network news protocol (nntp)
user	Regular user processes
uucp	UUCP subsystem

Question: What facility does SSH use? I want to have a dedicated log file for SSH, is this possible?

Answer: Because the SSH daemon uses authentication it would naturally use the facility auth by default. This can be changed by going to the SSH daemon's configuration file and instructing it to use another facility.

Question: Can facility be used by 1 subsystem at a time?

Answer: Multiple subsystems can share the same facility. A facility is a means of labelling an event generated and multiple subsystems may share that label (facility).

Question: Can additional facilities be created?

Answer: That is a yes and no answer. Yes, due to the nature of Open Source software, it is possible for you to create additional facilities but this is against the RFC for syslog (RFC 5424).

Priorities

Priorities are also predefined and are as follows:

Code	Severity	Keyword	Description	General Description
0	Emergency	emerg (panic)	System is unusable.	A "panic" condition usually affecting multiple apps/servers/sites. At this level it would usually notify all tech staff on call.
1	Alert	alert	Action must be taken immediately.	Should be corrected immediately, therefore notify staff who can fix the problem. An example would be the loss of a primary ISP connection.

2	Critical	crit	Critical conditions.	Should be corrected immediately, but indicates failure in a primary system, an example is a loss of a backup ISP connection.
3	Error	err (error)	Error conditions.	Non-urgent failures, these should be relayed to developers or admins; each item must be resolved within a given time.
4	Warning	warning (warn)	Warning conditions.	Warning messages, not an error, but indication that an error will occur if action is not taken, e.g. file system 85% full - each item must be resolved within a given time.
5	Notice	notice	Normal but significant condition.	Events that are unusual but not error conditions - might be summarized in an email to developers or admins to spot potential problems - no immediate action required.
6	Informational	info	Informational messages.	Normal operational messages - may be harvested for reporting, measuring throughput, etc. - no action required.
7	Debug	debug	Debug-level	Info useful to developers

			messages.	for debugging the application, not useful during operations.

Priorities represent the urgency of a message, and based on your configuration you are able to specify that informational messages go to particular log file while erroneous messages are logged elsewhere.

The higher the number, the lower the priority. This makes **debug** the lowest priority and **emerg** the highest priority. Listing a lower priority includes the priorities superior to it. Should one use the priority **err** to match to events then it would include **crit**, **alert**, and **emerg** too.

The configuration is done in **/etc/syslog-ng/syslog-ng.conf.**

syslog-ng uses the following elements for its logging framework:

Element	**Description**
source	This defines where syslog-ng listens for logging messages from.
destination	This defines where syslog-ng sends logging messages to.
facility	This represents a subsystem responsible for generating a message.
priority	This represents the urgency of a message.
filter	This is created by the administrator and consists of 1 or more facilities and priorities.

Let’s start by having a look at some configurations from **/etc/syslog-ng/syslog-ng.conf**

```
source src {
        #
        # include internal syslog-ng messages
        # note: the internal() soure is required!
        #
```


```
        internal();

        #
        # the default log socket for local logging:
        #
        unix-dgram("/dev/log");

        #
        # uncomment to process log messages from network:
        #
        #udp(ip("0.0.0.0") port(514));
};
```

This creates a source called **src** and listens for messages **internally** only. Should you wish to listen for syslog messages from another computer, a source for that computer needs to be defined.

```
destination mailinfo { file("/var/log/mail.info"); };
```

This creates a destination called **mailinfo** which sends messages to a **file** called **/var/log/mail.info**

```
filter f_mailinfo   { level(info)      and facility(mail); };
```

The above created a filter called **f_mailinfo** which represents the **facility mail** of the **priority info**. The facility and priority does not have to be individually referred to, instead **f_mailinfo** represents them both.

The actual configuration element which instructs syslog-ng to do the actual logging is the following line:

```
log { source(src); filter(f_mailinfo); destination(mailinfo); };
```

This says that we would like to log messages coming from a **source** called **src** (previously defined) concerning the **filter f_mailinfo** (previously defined) and we want it sent to a **destination** called **mailinfo** (also previously defined).

Without the above log line, no logging of events would happen and the other configuration elements would prove useless.

Testing a syslog configuration may be done from the command line using the **logger** command as follows:

```
$ logger -p mail.notice This is a message being sent using the facility mail and priority notice
```

Should your configuration be working, you should have the text "**This is a message being sent using the facility mail and priority notice"** in your logfile **/var/log/mail.info**

Logging Labs

Configure the SSH daemon to use the syslog facility **local4** and then create a rule in /etc/syslog-ng/syslog-ng.conf to ensure that all events using that facility are sent to /var/log/sshd.log.

Now head over to your virtual machine and configure the rsyslog daemon there to send events using the facility **local4** to your physical machine using TCP as the transmission method.

What port does rsyslog use by default?

Answer	

Is it possible for different facilities to be logged to the same file? If you're unsure, try it out.

Answer	

Can 1 facility be logged to different files? If you're unsure, try it out.

Answer	

BONUS:

Take events of the priority **debug** and log them to a file /var/log/debug.log but EXCLUDE the facilities **mail** and **kernel**.

Changing Kernel Settings

For those who wish to have systems that really perform well, we would offer this advice: Do not be content with the default settings. There are many settings that

will improve your system's performance. The shortcoming of administrators is usually a lack of familiarity with what can be changed.

/proc/sys and /sys

Linux has two pseudo-filesystems worth mentioning. /proc/sys has an accompanying configuration file (/etc/sysctl.conf) and /sys, which has no such designated file.

In /proc/sys, there are sub-directories that correlate to the resources that are managed through this system. /sys has block information for devices about how other resources work with disk.

/etc/sysctl.conf

To persistently change the settings under /proc/sys, there is a configuration file called /etc/sysctl.conf. Do not try to brave this document alone! Be sure to install kernel-doc, which contains a lot of kernel documentation. Included in this package are explanations about the settings under /proc/sys.

/etc/grub.conf

This is actually a link to /boot/grub/grub.conf. This is the configuration file for the bootloader. What this means is that any settings you wish to be passed to the system at boot time, when the kernel is loaded, can be persistently stored in this file.

Note: In the live environment this file does not exist.

Rather than speculate about this file, lets look at the kernel-doc files to see what settings can go in /etc/grub.conf. All of the options discussed can be appended to the "kernel" line of the grub.conf file.

/etc/modprobe.d/something.conf

Drivers sometimes have variables, or parameters, that can be set. Do see a list of drivers, or modules, run the command **lsmod**. Once you identify a module you would like to change run the command **modinfo <module name>**.

Within the /etc/modprobe.d/<module name>.conf file put the following:
options <module name> parameter=value

/etc/rc.local

This is the duct tape of the linux system. Anything that does not have a setting in a config file can be set here and you can have confidence that it will be applied each time the system starts or somebody changes runlevels.

kernel-doc

This package provides valuable information about kernel settings. It is always a good idea to install it.

```
# zypper install -y kernel-src
```

Then go to **/usr/src/linux/Documentation**

Changing Kernel Settings Labs

First verify that the xt_recent module isn't loaded: **lsmod | grep xt_recent**

Now load the module using the command: **modprobe xt_recent**
Verify that the module is loaded now.

Let's reboot our system.

Is the module **xt_recent** loaded now?

Answer	

To configure this behavior, we need to edit a configuration file.

Create a file called **/etc/sysconfig/modules/local.modules** with the following content:

modprobe xt_recent

Reboot your machine now and check to see if the module is loaded.

The files in /etc/sysconfig/modules will be processed under 2 conditions:

1. They end in .modules
2. They are marked as executable

Deal with these right now, reboot your system and verify that the module **xt_recent** is now loaded.

Let’s now disable icmp responses at a kernel level so that your system won’t respond to any ping requests.

Ping your own IP address right now and verify that your server responds to ICMP.

Now open **/etc/sysctl.conf** and add the following line.

net.ipv4.icmp_echo_ignore_all = 1

Once you have saved the above, return to your shell and verify that your system is still responding to ICMP requests.

Now reprocess /etc/sysctl.conf using the command: **sysctl -a**

Does your system now respond to ICMP?