Linux Academy.com

RHCSA 7

Study guide Part 1

Table of Contents

Prerequisites

Prerequisites

- The student has completed Linux Essentials on LinuxAcademy.com or has 1 year of Linux experience.
- The student has access on their local machine to virtualization software to run Red Hat for tasks that are unavailable in a virtual environment.
- Note: The student DOES NOT have to have access to Red Hat 7 or local machine to complete 90% of this course and prepare for the exam. The components that require a physical or virtual machine are limited in scope and will be covered enough for the user to be able to memorize and complete the requirements during the certification test.

Document conventions

- This document is designed to review the concepts already learned while taking LinuxAcademy.com RHCSA video lessons, labs, and exercises. It is meant to be used as a review and memorized before performing on an exam.
- Anything following a -> is a description or better known as a “comment” to a command just issued.
- Commands on the notes are displayed at a command prompt [user@localhost]$.
- Commands that should be issued as sudo or as ROOT will be denoted with the root command prompt [root@localhost]#.
- In these notes, we will assume that all commands to be executed as the root user will be executed by the root user and will not use sudo. Sudo is a best practice command in a production environment, however, to reduce clutter and confusion on the notes we will not be using it for commands.

Course Software

- LinuxAcademy.com Red Hat 7 Enterprise Lab server
- LinuxAcademy.com Red Hat 7 Enterprise with VNC (GUI) lab server
- Red Hat 7 physical machine for tasks that cannot be done virtually
- It is acceptable for the student to use CentOS 7 to complete and follow this course. CentOS7 is essentially a free fork of Red Hat Enterprise. This course uses Red Hat Enterprise 7 in training for 100% accuracy.

Getting started packages for the course:

- vim
- man-pages
- bash-completion

Usage Statements - Generally found in the man pages or help section of a program. Usage statements have their own syntax for understanding how to use programs.

- [ ] Surround additional “options” (For embedded brackets take out the outer most set of brackets for order)

- …   Means an arbitrary length of items of that specific type
- | When multiple items are separated by the "pipe", it means only **one** of them can be specified
- < > Text surrounded by these symbols represent it is variable data, the "parameter" for the command cd /home/user then /home/user would be variable data.
- cp, source/dest (minimum arguments)
- If you have one source file destination, it can be anything.
- If you have more than one source file, the destination needs to be a directory (not renaming).

Man

- [user@local]$ man -k tar   Search short descriptions for manual pages that include the word tar
- [user@local]$ man -k vim   Will return all man pages with a description that includes the text VIM; great to use if you cannot find a man page for an item.
- [user@local]$ apropos tar   Apropos command will search the descriptions AND the man pages for the given text. Man -k just searches the short descriptions.

Man pages are made up of sections. Each section contains a specific type of manual for a program or configuration file. Install additional Linux man pages from the Linux foundation; yum install man-pages.

Issue basic commands and navigate the system

- **pwd** - Show current working directory path
- **cd** - Change directory
- **ls** - List directory contents
- **sudo** - Sudo allows a regular user with sudo permissions to issue commands "as the root user".
- **mkdir** - Create a new directory
- **mkdir -p /create/all/dirs** - Will create all locations in the path if the do not already exist
- r**mdir** - Remove directory
- **rm -rf** - Remove directory that is not empty (recursively) and forcefully do it (assume yes at when asked)
- **Tab completion** - Tab completion can be used to complete "the next commands" or completion of the command name. File completion is also applies using tab tab.
- **rm {file1,file2,file3}** - Will remove file1, file2, and file3 all at the same time. The brackets {} specify multiple files or directories and can be used with different commands such as the touch command
- **touch -** Command can be used to create empty files, update the last modify time and access time to the current time of issuing the touch command against a file.

Input-Output Redirection (>, >>, |, 2>, $>, <, <<)

- **>** - Redirect standard output to filename
  - echo "test" > file.txt
  - Each time a single > is used; it will replace the existing file and contents if file.txt already exists.
  - ls -R /etc/ > /home/user/directory.tree
- **>>** - Redirect and append standard output to a file
  - echo "test" >> file.txt"

- echo “test1” >> file.txt”
- cat file.txt and both test and test1 will be there

- **|** - Chains scripts, files, and commands together by redirecting the stdout as stdin to the next command
  - cat /etc/passwd | grep root
  - ls /etc/ | grep motd > /home/user/file.out
- **2>** - Redirects standard error to a file
- **2>>** - Redirects and appends standard error to a file
- **/dev/null** - Empty space directory; any data sent here is immediately lost. Often used to pipe standard error or standard output from a script so that output does not clutter the terminal screen.
  - ls /etc/ > /dev/null
- **2>&1** - Redirects standard error to standard output
- **<** - Accept input from a file
  - mysql < filedump.sql
- **less** - Is a file viewing program that standard output can be piped into for easy browsing and searching of contents. Example: cat /var/log/syslog | less
- **head** - Shows the top 10 lines of a file.
  - -n specifies the number of lines to show at the top of a file. If the -n option is not passed then 10 is the default
- **tail** - Shows the last 10 lines of a file
  - -n specifies the number of lines to show from the bottom of the file. If the -n option is not passed then 10 is the default

File system Hierarchy Standard

Inside Linux, everything needs a home and an order to create a standard layout and home for common configuration files, programs, and locations on the system. Therefore, the File system Hierarchy Standard was created. Generally the common directories and directory usage should be the same across all Linux distributions. Below is a list of common directory locations on Red Hat and other distributions.

- /etc
  - Contains configuration files for a specific system. Generally these configuration files include programs and packages that have been installed after a system has been setup. Some configuration files are located in /usr/lib. However, generally if those configuration files are located in the /usr/lib directory there is a symlink located somewhere in the /etc directory tree linking to the system specific configuration file in /usr/lib.
- /var
  - Variable data specific to the system. This data should maintain persistence (should not be removed or changed) when the system is rebooted. Tools such as Rsyslog and Apache web server use this location for storing log files and website files to be served by the Apache web server. Some databases also maintain data store files in the /var directory. It is intended for data that is going to grow and shrink in size.
- /run
  - Runtime data for processes started since the last boot. An example of this runtime data would be the default configuration for systemd-journald service on Linux. The message information is stored

in the /run directory. This directory also replaces the /var/run /var/lock directories on older versions of Red hat. **All contents in the /run directory are either recreated or removed at boot time.**

- /home
  - Location where regular user home directories should be; used for storing personal information on the system.
- /root
  - Home directory for the root user.
- /tmp
  - Files in this directory will be removed if they are more than 10 days old. It has world read/write permissions so any program/user can write to this directory.
- /boot
  - Files needed in order to start the system boot process.
- /dev
  - This directory contains information on essential devices on the system. For example, contains information on devices that are attached to the system including, but not limited to, hard drives and USB devices.
- [root@localhost]$ man 7 hier - for more information, yum install -y man-pages needs to be installed for it to appear in the man pages.

Use Grep and Regular Expressions To Analyze Text

Grep - Grep is used to print lines that match a given pattern. The pattern can be regular text or made up of regular expressions patterns. A regular expression is an expression that describes a set of strings. Grep uses basic regular expressions by default but can be expanded to extended regular expressions with the -E option (man grep).

grep [pattern] [file]

grep jeff /etc/passwd
grep -i jeff /etc/passwd - Show all occurrences of "jeff" and ignore case specification (so jeff is the same as JEFF)
grep -v jeff /etc/passwd - Show all lines that do not contain the regular expression pattern "jeff". This is the "inverse".

grep can be used by itself when passed a pattern and file. However, you can also pipe standard output to the grep command for processing. Example: cat /etc/passwd | grep jeff

- grep linuxacademy filename - Search for lines in filename for the pattern linuxacademy
- grep '^linuxacademy' filename - Search for lines in filename that *start* a line
- grep 'linuxacademy$' filename - Search for lines that end with linuxacademy
- grep '^linuxacademy$' filename - Search for lines containing only linuxacademy in filename
- grep '[^abc]' filename - Will match any character not contained in the brackets

- grep [Ll]inuxacademy filename - Will search filename for any lines that have a pattern matching linuxacademy or LinuxAcademy
- grep ‘^&’ filename - Search for empty lines (starts with nothing ends with nothing)
- grep ‘a\{1,2\}’ filename - Will match either aa
- grep -v ^# filename -> List all lines that are not a comment #
- grep ^# filename -> List all lines that start with #
- -v - Selected lines are those not matching any of the specified patterns
- egrep is the same as grep -E: interpret pattern as an extended regular expression
- o fgrep is the same as grep -F: interpret pattern as a list of fixed strings, separated by new lines, any of which is to be matched
- -i -Ignores case and matches any single character
- [^] -Matches any character not contained in brackets; [^abc] matches any character other than a, b, or c
- ^ -Matches the starting position of a line
- $ -Matches the ending position of a string or the position just before a string ending new line it matches the ending position of any line
- * -Matches the preceding element 0 or more times ab*c matches b zero or more times. IE ac, abc, abbbbc, abbbbbbbbbbbbc.
- . -Matches any single character
- ? -Preceding item is optional and is matched at most one time
- * -Preceding item will be matched 0 or more times
- \+ -Preceding item will be matched one or more times
- (123){2} -Matches 123123
- {n} -Preceding item is matched exactly n number of times
- {n,} -The preceding item is matched n or more times
- {,m} -The preceding item is matched *at most* m times.
- {n,m} -Preceding item is matched at least n times but not more than m times
- In basic regular expressions, the meta-characters ?, +, {, |, (, and) lose their special meaning; instead use the backslashed versions \?, \+, \{, \|, \(, and \).
- *man grep*

Access Remote Systems Using SSH

Password authentication allows a local user to login to the system with just a password. This is considered to be a less secure method. Alternatively, key based authentication can be enabled. Key based authentication requires an RSA or DSA key to be generated. The public key is to live on the server accepting incoming connections while the private key needs to exist in the proper location of the connecting client in order for key signing and verification to be successful. It is bad practice to allow root user to login to the system. The server knows the public key, and only the user knows the private key. You should always use “sudo” or su - into the root user and not enable root login directly. By default this is disabled on Red Hat 7.

ssh [servername] [command] - Allows you to issue commands on a remote server without connecting

ssh [user@servername] - Allows you to connect to a remote system with password authentication enabled

SSH configuration file is located in /etc/ssh/ssd_config

When SSH is enabled, you can connect to it multiple ways- through terminal as well as through SFTP and SCP.

- scp [filename] [user@servername:~/] - ~/ represents the users home directory; location to upload file must be specified.
- sftp [user@servername]
    - ? -To display list of available sftp commands
    - ls -to list file contents
    - cd -to navigate the directory
    - get -to download
    - quit or by -to exit sftp
    - put local filename remote location (local file name assumes you are in the current working directory)
    - get filename (to download a file from the sftp server)

Note: SSH Security Keys are addressed in the security section of the course and in the "advanced study guide" as part of the RHCSA downloadable course material. It is not covered in this specific study guide.

Login and Switch Users in Multi-user Targets

Targets are systemd unit configuration files used for grouping together a set of resources. For example, a graphical target will group together systemd unit configuration files that start graphical user interfaces such as X server. Multi-user target is a target that enables multiple users to be logged into the system and is generally the default target for a system. *Targets will be covered in-depth later in this course.*

An **interactive shell** is any shell that has a prompt that allows user interaction on the system. A login shell will load the profile customization files for a given user and the global files. Just changing users using the su command does not enact a login shell. If you change a user by typing su user, then you are logged in as the user "user" but only in an interactive shell and not a login shell. All the user's customizations and environment variables that are loaded are not loaded because a login shell was not specified. This will cause substantial issues if scripts are created for individual user information. Such as running a script as root but not using a login shell to change into the root user.

- Change users in a multi-user target
    - su username
    - su - username, the - creates a login shell. su - login username will also create a login shell as that user

GNU Bourne-Again Shell (Bash) Improved version of the Bourne Shell (SH)

- Interactive Shell - Generally displays a $ prompt when it is used interactively. As the root user, the $ is removed and # is instead displayed .
- Commands - Commands are names of programs to be run. They are generally followed by "options",

more commonly known as “flags”. The flags provide additional uses for the program being called.

- Commands at the prompt are made up of three parts:
  - A command to run
  - Options to pass the command, also known as flags. Options enable the ability to specify different use cases for a command or expected behavior.
  - Arguments of the command

Archive, Compress, Unpack, and Uncompress Files Using tar, star, gzip, and bzip2

Tar - Tar is an archiving utility. It does not handle compression but instead archives directories and files into a single file labeled as .tar. Most of the time when users need to compress directories and files together they first archive it in a tar and then compress it using a compression utility.

- -c create a new archive
- -t list contents of the archive
- -x extract files from the archive
- -z compresses in gzip or decompresses depending if -x or -c flag is passed
- -v display the files verbosely
- -j filter through bzip2
- -f use archive file (filename passed)
- Examples
  - [user@local]$ tar -cf hello-world.tar hello world -> creates a hello-world.tar archive that contains both hello and world files
  - [user@local]$ tar -tvf hello-world.tar -> list all files in the hello-world.tar archive
  - [user@local]$ tar -xf hello-world.tar -> extract files in the archive
  - [user@local]$ tar -czvf hello-world.tar.gz file1 file2 -> create a tar out of file1 and file2 and then compress the archive
  - [user@local]$ tar -zxvf hell-world.tar.gz -> extract the files from the archive but first unzip the file compressed with gzip

star - Star is an archiving utility (not installed by default) similar to tar but with a few key differences. Star is generally used to tar large data sets of information. It includes utilities that allow for pattern matching and searching in order to create, unpack, and list the contents of a star archive. Star includes a larger set of utilities included in the program specifically designed for working with archives.

To install star **[root@local]# yum install star**

- Options/flags
  - -c -> creates a new tar file and writes the names to it.
  - -v -> show actions verbosely
  - -n -> show what “would” happen if you executed the star command but do not extract it
  - -t -> list the contents of the tar file
  - -x -> extract the specified named contents of the tar file

    - --diff -> show differences between named files and the file system
    - -C -> change to specified directory before executing command
    - -f=filename -> specify the tar file name to be created
- Examples
    - [user@local]# star -c -f=archive.tar file1 file2 -> Create a star archive named archive.tar and include file1 and file2
    - [user@local]# star -c -C /home/user -f=archive.tar file1 file2 -> Change into the /home/user and then create archive.tar and include /home/user/file1 and /home/user/file2
    - [user@local]# star -x -f=archive.tar -> Extracts the archive.tar file by default when using -x star will not extract any file which is older than the corresponding file on disk in the current location
    - [user@local]# star -t -f=archive.tar -> List the contents of archive.tar
    - [user@local]# star -t file=archive.tar file1 -> Will extract file1 and only file1 from the archive

gzip - Gzip is a compression utility used to compress file sizes to reduce the amount of space required. When compressed, the files are unavailable until they have been unpacked for usage. Generally gzip is used with an archiving utility like tar or star. In order to compress directories, you must first place the directory in a archive.

Gzip is installed by default on Red Hat 7.

- Options/Flags
    - -d decompress
    - -l list compression information about each compressed files
- Examples
    - gzip file1 -> Compresses file1 into file1.gz
    - gzip -d file1.gz -> Decompresses (unpacks) file1.gz
    - gunzip filename -> Also decompresses (unpacks) a compressed archive

Create and Edit Text Files

vi - Vi is an older editor that has been replaced by Vi iMproved editor. VIM light is now known as vi and is installed by default on a system. It is important to learn this editor because in recovery modes, vi editor is always installed and useable while other texted editors will not be available.
vim - VI iMproved, the full version of VI which brings a lot of additional functionality
nano - A simple text editor used for editing files
touch - Creates or “touches” files modifying the timestamps accordingly; this is often used to create empty dummy files.
echo “text” > file.txt - Will create a file called file.txt that includes the word text
echo “text” >> file.txt - Will append text to file.txt if it already exists. If it does not exist, it will create file.txt.

Important: This course will be using vi/vim throughout for examples. It is encouraged that you learn and also use this editor. Please see the required reading section of the Red Hat 7 RCHSA course and download the VI study guide that includes all information needed.

Create, Delete, Copy, and Move Files and Directories

mkdir - Makes directories

- mkdir -p /dir/dir1/dir2 -> -p flag will create the parent directories dir,dir1 and then create dir2 if they do not already exist. If dir exists but dir1 does not, it will create dir1 and then dir2.
- mkdir /dir/dir2 - If dir does not exist, then creation of dir2 will fail unless the -p flag is passed in to create the "parent" directories.

cp - Used to copy files and directories

- cp file1 file2 -> Copies file1 and creates file2.
- cp -R dir1 dir2 -> Copies dir1 and creates dir2. -R means recursive so it will copy the directory and all files under the directory. This is required in order to copy directories.

mv - Move files. This is also how you "rename" files in Linux.

- mv file1 /etc/file1 -> Moves file1 from current location to /etc/file1.
- mv /etc/file1 /etc/file2 -> Moves /etc/file1 and renames it to file2 by moving it into /etc/file2.

rm - remove files or directories

- -r or -R -> Remove directories and all their contents recursively
- -f -> force, do not prompt for verification of removing file
- -i -> prompt before removal (alias rm)
- examples
    - rm -rf dir1
    - rm -i dir1
    - rm file1
    - rm -f file1
    - rm -rf file1,file2,file3
    - rm -rf file*
    - Expressions can be used to pass in specific file names. rm file* means delete all files that start with "file" and have any sequences of characters after. For example: file1, file2, filefilefile, fileisthis, file_1, etc.
    - rmdir directory - Will remove an empty directory. rm -rf dirname is required to remove a directory with contents in it.

Create Hard and Soft Links

The ln command is used to create links between files. Think of a link similar to a shortcut on a desktop to a file on a system. There are two types of links. **Soft links**, also known as **symlinks** or **symbolic** links, make a symbolic link to a file. Every file on the system by default has one hard link. A **hard link** links directly to an inode and is just a new entry with a reference to an existing file on a system.

Symbolic/Soft Links

ln -s /etc/motd mymotd -> Creates a symbolic link to /etc/motd named mymotd.

Considerations for symbolic links:

- Symbolic links can be broken if the source or target is moved or removed.
- When you edit a symbolic link, you are essentially opening the source location file for editing.

- You can have multiple symlinks to the same file on a system.
- Symbolic links can link across file systems.
- Permissions on a symbolic link do not matter; instead the source permissions on the target file is what the system reads when opening the link.

Hard Links
ln /etc/motd mymotd -> Creates a "hard" link to /etc/motd named mymotd.

Considerations for hard links:

- Hard links link directly to the inode source on the hard drive.
- If the source/target file is removed, the created link still exists and the data is available on the inode until all links are no longer available.
- All hard links linking to the same file will have the same date times as well as permissions.
- After a new hard link is created, it is not possible to determine which one was the original file.

Viewing Symbolic and Hard links:

- ls -l will display information about symlinks and hard links.

List, Set, and Change Standard ugo/rwx Permissions

There are two ways to set permissions on standard Linux files and directories. The standard symbolic ways use characters such as u,g and o to define what permission you are setting and r, w, and x to define the permission level. The chmod command is use to set these permissions. The second method is using octal bits such as 777 and 755 to represent and set permissions on a file and directory. For the RHCSA exam requirements, it is only required that we know the standard ugo/rwx. However, we do spend time in the videos covering octal permissions as well.

Each file or directory has owner/group/other permission level. Owner is the permission level for the owner of the file, group is the permission level for the members of the group who owns the file, and other is for every one on the system. Other is generally considered a security issue depending on what files are set to other.

chmod - Short for "change mode", is used to change the file mode bits of the given file.

- u - user
- g - group
- o - other
- a - all
- r - read
- w - write
- x - execute
- s - setuid or setgid

-X - (used with the -R recursive flag on chmod) is used to indicate that execute permissions should only be set on directories not regular files. This ensures that users can navigate into the directory but does not give execute permission to files/scripts.

s - set user or group id for execution, when set on directories then files or directories created will inherit the parent (the parent directory being the directory with the set-GID bit on it) directories group ownership. Set-UID on a file will have the file be run with the permissions of the user or owns the file not of the user running the file. This is common on commands such as the “passwd” command. For security reasons it can no longer be used on bash scripts.
t - prevent user from deleting, aka sticky bit

- On directories, if the sticky bit is set, it will prevent unauthorized users from removing or renaming a directory and a file in the directory unless they own the file.

Examples:

- chmod ug+rw filename -> Add read and write permissions to filename for the user/group owner of the file
- chmod a+r -> Add the read switch to everyone
- chmod -R g+rX dir -> Recursively set read permissions on all files and directories starting in dir recursively and set execute permissions on directories only.

Octal Permission Bits
1 = Execute
2 = Write
4 = Read

chmod 0000 filename -> First column is for special permission bits. Second, user owner permissions. Third, group owner permissions. Fourth, world/everyone/other permissions.

- Octal bits are set by adding the desired permissions together.
- Often the first digit is left off if the permission bit is not set.
  - Examples
    - Set user owner permissions to read write and all others to no permissions
      - Read = 4, write = 2, thus 4+2 = 6, **chmod 600 filename**
    - Set user owner and group owner to read/write/execute and read only permissions for all other
      - **chmod 774**
        - User owner -> 4+1+2 (read + execute +write) = 7
        - Group owner -> 4+1+2 (read + execute +write) = 7
        - Other -> 4+0+0 (read + execute +write) = 4

Chown - Used to change user owner and group ownership of a file or directory.
chown username:groupname file

- The position before the : represents the user that owns the file
- The position after the : represents the group that owns the file
-

chown -R username:groupname directory -> Set ownership recursively
chown username file -> set user ownership on a file
chown :groupname file -> set group ownership on a file
chown username:groupname file -> set both user and group ownership on a file

chgrp - Can also be used to change the group ownership. Example [root@localhost]# chgrp groupname file|dir
newgrp - Log into a new group, changes the current users primary group.

setuid - Setting the setuid permission bit on a file that is executable means that when the fie is executed then it will be run with the same permissions as the user who owns the file, not the user who runs the file. Generally, this method is restricted on newer distributions. However, examples of commands that have this set is not limited to but includes passwd.

setgid - Setgid is the same as setuid but for group instead.
When running the file, it causes the resulting process to be owned by the owner of the group and not of the user who read it. When set on a directory all sub files/directories will inherit the group ownership of the parent directory (the parent directory being the directory with the set-gid bit enabled).

Files
chmod u+s filename -> File will execute as the user who owns the file not the user who executes the file.
chmod g+s filename -> File will execute as the group who owns the file not the group that executes the file.

Directories
chmod g+s dir -> setguid effects directories because files that are created inside of a directory with setgid/setuid set will have their group owner set to match the group who owns the directory.
chmod u+s dir -> setuid does not effect directories

* Sticky permission bit (chmod o+t) -> Users that write to a directory can only remove files that they own. They cannot remove files they do not own regardless of permissions.

umask is used for setting the default file permissions for new directories and files
The permissions for a file are set by the process that creates them. For example, a text editor creating a file. When a text editor creates a file, it creates it with the permissions read/write. The text editor does not need execute permissions so they are not created. Umask, "masks" or "hides" those default permissions. For example, when you "touch" a file, the touch command/process does not need execute permissions. If umask was not set, it would be created with rw-rw-rw.

- umask (aka user mask)
  - An easy way to determine the umask is to subtract the value of the permissions you want from 666 (if for a file) and 777 (for a directory).
  - Displays current value of shells umask
  - Umask "masks" or "hides" permissions
  - umask uses octal bitmask to clear permissions on newly created files and directories
  - When using the umask command, changes will apply only to the currently shell.
  - **For persistence,** modify the umask settings in /etc/profile and /etc/bashrc files.
  - Umask "masks" permissions on files.
  - Can be set with symbolic and octal "bitmask" notation
  - If three characters are passed, then a leading zero is assumed.

- Examples
  - **Mask permissions for "other" users to write a file when created AND this also masks "other" permissions from having write access in a directory. Note that a directory needs x in order to**

switch into a directory, so umask behaves differently on directories than it does files.
- umask 022
  - **Mask write access for group members and all of the "other" permissions**

Locate, Read, and Use System Documentation Including man, info

- command --help
- info
- each info page is referred to as a node and the top line of the node is referred to as the **mode line**
  - has next/previous/up (not at all times)
  - to move to the next node, hit the **n** key
  - h -> while in info to learn how to use it
  - nn -> brings up advanced commands
  - info --apropos=string -> This can be used to search the info pages.
- **which** - Shows the full path of a command. Commands have executable programs located on the system. In order for these commands to execute without typing the full absolute path, the location of the executable is generally in the users $PATH environment variable. If you ever need to know the full absolute path to a program/command in order for scripting purposes or general knowledge, which will display that information.
- **whatis** - display manual page descriptions
- **locate** - locate or "find" files on a system by a given name. Locate searches a cached database of files that is frequently updated. Having a cached database results in a faster search. However, data that might have been recently added/changed may not be updated in the system. See updatedb.
- **updatedb** - Updates the database that the locate command uses. The command is usually run **daily** by cron but can be run manually.
- **man**
  - Program/Unix modules information are located in the man program. The man program is made up of 9 different sections listed below:
    - 1 Executable programs or shell commands
    - 2 System calls (functions provided by the kernel)
    - 3 Library calls (functions within program libraries)
    - 4 Special files (usually found in /dev)
    - 5 File formats and conventions (such as /etc/passwd /etc/shadow)
    - 6 Games
    - 7 Miscellaneous
    - 8 System administration commands (commands usually reserved for the root user)
    - Kernel Routines
  - Examples using the man command
    - man passwd (passwd is an executable program or shell command so it opens under man page 1)
    - man shadow (Will open the first found manual which is section 3, a library call for programmers. However, shadow is also for /etc/shadow file in etc.)
    - man 5 shadow (Will open the 5th section of the man program for shadow specific

information. Since this is file formats, it will open information about our /etc/shadow configuration file.)

- **info** - Info is a utility for reading information files. For some Linux commands, it provides more detailed information and examples.
- **apropos tar** -> Apropos command will search the descriptions AND the man pages for the given text. Man -k just searches the short descriptions.

Info manuals
/usr/share/info (all info manuals are located here)

When an info page is not available, it will default to the man page.
/usr/share/doc

- info file navigation commands

Keystroke	Action
?	Displays help information
N	Moves to the next node in a linked series of nodes on a single hierarchical level. This action may be required if the author intended several nodes to be read in a particular sequence.
P	Moves back in a series of nodes on a single hierarchical level. This can be handy if you've moved forward in such a series but find you need to review earlier material.
U	Moves up one level in the node hierarchy.
Arrow keys	Moves the cursor around the screen, enabling you to select node links or scroll the screen.
Page up, Page down	These keys scroll up and down within a single node, respectively.
Enter	Moves to a new node once you've selected it. Links are indicated by asterisks (*) to the left of their names.
L	Displays the last info page you read. This action can move you up, down, or sideways in the info tree hierarchy.

T	Displays the top page for a topic. Typically this is the page you used to enter the system.
Q	Exits from the info page system.

Boot, Reboot, and Shut Down a System Normally

Rebooting a system in Red Hat 7 is a little bit different. All the previous commands are kept for backwards compatibility. However, all of the shutdown functions are now handled by the systemd daemon. Here are a few ways to boot, reboot, and shut down a system normally.

- **Rebooting the system**
  - Reboot
  - systemctl reboot
  - shutdown -r now (-r means reboot)
    - now -> reboot immediately
    - +5 -> wait 5 minutes and then reboot
    - +0 -> means the same thing as now
    - 01:01 -> 1:01 am shutdown
    - -c -> will cancel a scheduled shutdown
  - init 6 (the init system in Red Hat 7 is deprecated. However, runlevels are still compatible for this current version for backwards compatibility)
- **Shutdown the system (no reboot/poweroff)**
  - systemctl halt
  - halt
  - shutdown -h now (-h means halt)
  - init 0
- **Physically power off the system**
  - systemctl poweroff
  - poweroff
  - shutdown -P

Boot Into Different Targets Manually

Systemd Overview*: Systemd has replaced the traditional SysVinit initialization system on Red Hat 7 distributions. The concept of run levels are still available but only for backwards compatibility. Instead, what has been introduced is the concept of “targets”. A target is a type of sytemd unit of configuration that defines a grouping of services or systemd unit configuration files that need to be started when the system “moves” into the defined target. In order for a system to be able to boot into a different target, the target has to be* *specifically configured. If it is not configured, the target is just used for grouping of configurations. This means the multi-user.target that the system can boot into can call the network.target. Where the network.target will have a grouping of .service unit configuration files that are needed for networking to be started on the system. However, even though the multi-user.target can call the network target, the system cannot "change into" the network.target because the network.target needs that special configuration.*

A target is just a grouping of dependencies that should be started when the target is called.

```
[root@localhost]# systemctl list-dependencies graphical.target
```

View all available targets on the system

```
[root@localhost]# systemctl list-units --type=target
```

View all targets installed on a disk

```
[root@localhost]# systemctl list-units --type=target  --all
```

Common Targets The System Can Be Booted Into

- emergency.target - su login only and mounts the / as read only file system
- multi-user.target - supports multi users being logged in the system and is a terminal (text) only; most common for servers
- rescue.target - su login prompt basic systemd init completed
- graphical.target - supports multiple users and generally launches or support graphical user interfaces

systemctl get-default (shows current target)
systemctl set-default multi-user.target
Note: the default target is configured as a symlink located in /etc/systemd/system/default.target

Systemd Confiuration files
/usr/lib/systemd/system
/etc/systemd/system -> files in here will overwrite the files with the same name in /usr/lib/systemd/system

View available systemd unit configuration types

- systemctl -t help

Find the status of a service or unit configuration file

- systemctl status unit.service
- *Note: if unit.service (the.service) is not passed, then the default assumption of systemctl is that it is a .service configuration file*

Systemctl commands

- systemctl --type=service (list all active service unit configuration files)
- systemctl is-active servicename
- systemctl is-enabled servicename
- systemctl list-units --type=service --all (list all unit configuration files regardless of if they are active or

not)
- systemctl enable service.service (enable the service configuration file to start at boot time)
- systemctl --failed --type=service (list all failed services)
- systemctl list-unit-files --type=service (view the enabled and disabled settings for all units of the type serivce)
- systemctl list-units --type=service
- systemctl list-units --type=service --all
-

Tip: You will notice that by default when you are issuing systemctl commands it will show only the loaded configuration files unless you pass the --all option.

Selecting a different target at boot time

1. Reboot the system
2. At the grub menu, make sure to stop the countdown to auto select by hitting any key
3. Hit the “e” key to edit the entry
4. Navigate down to the linux16 kernel command line and hit the “end” or crtl+e key to go to the end of the line
5. Add systemd.unit=rescue.target (replace rescue.target with the desired target)
6. Hit crtl + x

Man pages to assist in the exam:

- systemd.unit
- systemd.service
- systemctl
- systemd

Interrupt the Boot Process in Order to Gain Access to a System

1. Start or reboot a system to get to the boot menu
2. Press any key to stop the auto selection of a grub item
3. Ensure the kernel you intend to boot into is highlighted and press the “e” key to edit the entry
4. Navigate to the linux16 kernel line and hit the “end” key to go to the end of the line
5. Append **rd.break** to the linux16 kernel line
6. Hit crtl + x to continue
7. The system will boot into an emergency mode that has the /sysroot directory mounted as read only
8. Mount the /sysroot directory with read and write
   - mount -oremount, rw /sysroot
9. Switch into chroot jail and set the /sysroot as the root file system
   - chroot /sysroot
10. Reset the root password
    - passwd root
11. Cleanup -> Make sure that all unlabeled files get relabeled during the boot process
    - touch /.autorelable

12. Exit the chroot jail
    - exit
13. Exit the initramfs debug shell
    - exit

Note: See the LinuxAcademy.com video for information about autorelable or read CentOS documentation related here: https://www.centos.org/docs/5/html/5.2/Deployment_Guide/sec-sel-fsrelabel.html

Identify CPU/Memory Intensive Processes, Adjust Process Priority with renice, and Kill Processes

Note: This objective can be completed entirely with the top command. As part of this objective we will explore a few additional commands on the command line as well.

Nice Priority:

- -20 (Most favorable to the process)
- 19 (least favorable to the process)
- Any user can set a niceness of "lower priority or a higher niceness", but only a privileged user can run a program for lower niceness aka higher priority.
- [root@localhost]$ renice -n 5 $(pgrep httpd) - Set the niceness on all httpd processes to 5.

Top - Tool used for displaying, killing, setting the nice value, and viewing system usage from a process view

While running the top program, the following keys and shortcuts can be used:

- k - Kills processes
- q - Quits the top program
- r - Change process priority (renice)
- s - Change update rate of top program
- P - Sort by CPU usage
- M - Sort by memory usage, show uptime information, memory information, and load average
- l - Toggle load average display
- t - Toggle task display
- m - Toggle memory display
- B - Bold display
- **u** **-** Filter based off of username

Options passed to the top command at start time

- Start the top program, update 2 times, and exit
    - top -n 2
- Start in batch mode
    - top -b

- Delay 2 seconds between each update
    - top -d 2

Reading the top program columns

- PID - Process id
- USER
- PR - Priority
- RES - The non-swapped physical memory a task has used
- SHR - Shared memory size (the amount of shared memory available to a task, not all of which is typically resident. It simply reflects memory that could be potentially shared with other processes.
- %CPU - The tasks' share of the elapsed CPU time since the last screen update
- %MEM - A tasks' currently used share of available physical memory
- TIME+ - CPU Time - Total CPU time the task has used since it started

pgrep

- List all processes associated with a user
    - **pgrep -u username**
- List all processes associated with a user and display the process name
    - **pgrep -l -u userame**
- Kill all processes started from a specific terminal
    - **pkill -t ttyid**
- Show all processes that DO NOT belong to a user (inverse)
    - **pgrep -v -u username**
- Show the process id of the most recent process started for a user
    - **pgrep -n**
- Show all SSHD processes associated with a user
    - **pgrep -l -u user sshd**
- Kill all httpd processes
    - **pkill httpd**
- Kill all SSHD processes for a specific user
    - **pkill -u username sshd**
- Logout a user
    - **w (to list the tt/y)**
    - **pkill -t**
- Kill all processes started from a specified terminal
    - **pkill -t tty**

Kill signals (A few of the most commonly used signals): (**note: kill -l will show all available kill signals)**

1	SIGHUP	Hangup	Used to configure reload without termination; also used to report termination of a controlling process of a terminal
2	SIGINT	Keyboard interrupt	Causes a program to terminate. This is the same as using ctrl + c at the terminal to stop a program.
3	SIGQUIT	Keyboard quit	The SIGQUIT signal is sent to a process by its

			controlling terminal when the user requests that the process **quit** and perform a core dump.
9	SIGKILL	Kill, Cannot block	The SIGKILL signal is sent to a process to cause it to terminate immediately (**kill**). In contrast to SIGTERM and SIGINT, this signal cannot be caught or ignored and the receiving process cannot perform any clean-up upon receiving this signal.
15 (default)	SIGTERM	Terminate	The SIGTERM signal is sent to a process to request its **termination**. Unlike the SIGKILL signal, it can be caught and interpreted or ignored by the process. This allows the process to perform nice termination releasing resources and saving state if appropriate. SIGINT is nearly identical to SIGTERM.
18	SIGCONT	Continue	The SIGCONT signal instructs the operating system to **continue** (restart) a process previously paused by the SIGSTOP or SIGTSTP signal.
19	SIGSTOP	Stop, Cannot block	The SIGSTOP signal instructs the operating system to **stop** a process for later resumption.
20	SIGTSTP	Keyboard stop	The SIGTSTP signal is sent to a process by its controlling terminal to request it to **stop temporarily**. It is commonly initiated by the user pressing control-Z. Unlike SIGSTOP, the process can register a signal handler for or ignore the signal.

PS
- ps -e
- ps -U root (view all processes owned by root)
- ps axo pid,comm,nice

Locate and Interpret System Log Files and Journals

- Systemd comes with builtin support for a new daemon called journald; man pages can be found at man systemd-journald.
- Journald is responsible for event logging in the system; it will record events from log files, kernel messages, and more.
- By default journald does not keep persistent data after a reboot.
- Non-persistent journald journals are stored in the /run/log/journal directory.
- If the /etc/ /journald.conf file is configured for persistence, then the journald log files are stored in /var/log/journal.
- See man journald.conf for configuration of the systemd journal.
- The journal can be viewed in several ways, primarily using the Journalctl command.
- journalctl
    - **-n** show "n" number of last lines appended; if no number is added, then the default is 10

- **-x** provide additional explanation texts from the message catalog if available
- **-f** show the last 10 events and continue listening
- -b will show messages from this boot only
- -b-1 will filter out to show the previous boot error messages
- **journalctl -b-1 -p err** - show previous boot error logs
- **journalctl _SYSTEM_UNIT=httpd.service** - to get all events related to the httpd unit, you can use this on any unit configuration type
- **journalctl --since=yesterday**
- **journalctl --until=00:00:00**
  - yesterday, today, tomorrow are all accepted with the --since and --until options as well as actual dates AKA 2012-10-30 18:17:16
- **journalctl -p err** - display all events with a syslog priority of err
- **systemctl status httpd** - shows the status of a service and most recent journald events

- Finding information about system bootup
  - **systemd-analyze** - Provides information about the system boot process
  - **systemd analyze-blame** - Provides information about each unit configuration during the boot process

List, Create, Delete Partitions on MBR and GPT Disks

- **fdisk** - Used to create MBR based partitions. It's important to remember to label the partition for the specific use case, be it Swap, LVM, or Linux file system.
  - MBR partitions can have only 4 primary partitions and each partition can have up to 2TiB in size. This is a real world limit these days.
- **gdisk** - Used to create GPT based partitions. GPT partitions are a UEFI standard. GPT partitions support 128 primary partitions on a device and each partition can have up to 8ZiB in size.

Create and Remove Physical Volumes, Assign Physical Volumes to Volume Groups, and Create and Delete Logical Volumes

Logical Volume Manager Process

1. **Physical Volume** Each LVM logical volume has an underlying physical storage unit. This is the physical volume component of LVM. A physical volume can either be a partition of a device or the entire disk. In order to use a physical volume, the physical volume must be initialized as a physical volume A.K.A. (PV). By doing this, a label is placed in the first part of the volume to help identify and provide meta data about the physical volume to our LVM manager. It is placed in the second 512-by sector of the physical volume. 0, 1, or 2 copies of this data can be stored on each physical volume; the default is 1 copy. Once configured, you cannot change the number of copies available. We know the first copy is stored at the start of the device but the second copy is stored at the end of the device. This helps protect against accidental overwriting of data.
2. **Volume group** is a combination of physical volumes that creates a pool of space that the LVM or logical volumes can allocate. Extents - Inside of a volume group, disk space is provided inside of allocatable

fixed units called “extents”. An extent is the smallest unit of space that can be assigned to a volume group. Volume group extents are referred to as physical extents and a logical volume is allocated into sets of logical extents that are the same size as the physical extents. Thus, the logical volume extents map to the physical volume extents and that is how a logical volume communicates with the physical volume data.

LVM Setup Process

- Once you have partitioned your volumes, you need to create a physical volume for LVM.
- pvcreate
  - pvcreate /dev/disk /dev/disk (you can list multiple volumes at one time to create)
- pvdisplay
  - Allows you to view the physical volumes that are available for usage by LVM
- Create a volume group
  - vgcreate vg-name /dev/ /dev (the disks that make up the volume group)
- vgdisplay
  - Allows you to view the available volume groups
- Create the logical volume
  - lvcreate
    - -n volume
    - -L size of the volume in bytes, i.e 1M 1G 20G 1M
    - -l size in physical extents (if you would rather specify this way)
      - If a physical extent is 4MiB (and by default it is) and you want to create an 8MiB LVM, then the -l option would be 2.
    - examples
      - lvcreate -n nameofthelvm -L 20G volumegrouptouse
    - View created LVM
      - lvdisplay
- Remove a logical volume
  - lvremove /dev/volumegroup/volume
- Remove a volumegroup
  - vgremove volume-group-name
- Remove physical device
  - pvremove /dev/device

Configure Systems to Mount File Systems at Boot by Universally Unique ID (UUID) or Label

After attaching a device and creating a partition, in order to mount the device you will need to format it with a filesystem.

- **mkfs -t xfs /dev/xvdf1**
- **blkid** - After a device has been formatted with a file system, this will list available block devices on the system.

- **lsblk** - Will list all the attached block devices.

A partition can be mounted at the command line, non persistently, with the following commands; assume the partition name is /dev/xvdf1

- mount /dev/xvdf1 /mnt/mountlocation
- Mount options can be passed to the mount command

Mounting with a UUID always ensures that the proper partition is mounted.

- mount -u 679ea5a2-c652-458a-8726-6e3970d1f58f /mnt/mountlocation
- Persistently mount the device in /etc/fstab
    - UUID=679ea5a2-c652-458a-8726-6e3970d1f58f /mnt/mountlocation defaults 1 1

Mount with the file system label: A file system needs to have a label in order for this to work. Below will show you how to create a label on both xfs and ext based file systems.

- ext based file systems
    - **tune2fs -L labelname /dev/xvdf1**
    - or **e2label /dev/xvdf1 lablename**
- XFS based file systems
    - **xfs_admin -L labelname /dev/xdf1**

Mount the file system based off label name.

- Edit the /etc/fstab file to create a persistent mount
    - **LABEL=labelname /mnt/mountpoint defaults 1 1**
- Mount at the command line non persistently
    - **mount LABEL=labelname /mnt/mountpoint defaults 1 1**

Additional Commands

- Attempt to mount all entries in the /etc/fstab file
    - **mount -a**
- Attempt to unmount all entries in the /etc/fstab file
    - **umount -a**

Schedule Tasks Using at and cron

at is a utility used to execute a command at a later time. You would use at to schedule a task to be run one time and use cron to schedule a task to run on a regular interval schedule.

- /etc/at.allow - Users listed here will have access to the at utility, all else will be denied.
- /etc/at.deny - Users listed here will not have access to the at utility, all else will have access. This is used instead of at.allow.

- at commands or a file which will be executed “once” at a later time.
    - at now +1 minute
    - at > echo “hello"
    - at >(crtl +d) to close the program
- Accepts the following time formats:
    - hh:mm - If the time specified has passed for the current day, it will assume it is meant for the next day.
    - midnight
    - noon
    - teatime (4pm)
    - Can also specify am/pm
    - Can also specify full dates or dates such as now +1 day, now +1 year, etc.
- atrm - atrm #job# will remove the job of the pending at task.

Anacron

Anacron is used to execute commands periodically, with a frequency specified in days. Anacron will look at the last time a command was run and determine if it has been run within the specified interval timeframe. This is different than cron because cron will run it on “given times of the month” and assumes your server or computer is always on. With cron you will set a time to run, but this will present an issue if your machine is offline or down during that time frame. Think of it as if you were running a laptop that ran a cleanup script. You’ve setup this cleanup script to run in cron on the 5th day of each month. However, your laptop is not running on the 5th of the month. This will cause the cleanup script not to be run that entire month. However, with anacron you can instead set it to run after a specified interval. When anacron runs, it will look to see if the script was executed in the last n (or say 30) days. This reduces any issues caused by systems that are down during regular cron schedules.

- -f Forces execution of all jobs, ignoring any timestamps
- -u Updates the timestamps of all jobs to the current data but does not run the jobs
- -n Run jobs immediately and ignores specified delays in the /etc/anacrontab file
- -t Uses a specified anacron config file rather than /etc/anacrontab
- -h Shows help
- /var/spool/anacron -> configuration directory that shows all timestamps for the anacron jobs
- Only system administrators can configure anacron and no user configuration is available

- **syntax**
    - period in days = specifies the frequency of execution of a job in days
    - delay in minutes = the number of minutes anacron waits, if necessary, before executing a job
        - 0 minutes means no delay
    - job-identifier - unique name of a job which is used in the log files
    - command - variable specifies the command to execute
- **Additional configuration variables**
    - start_hours_range - specific time frame in hours that anacron jobs can be run
    - random_day - sets a specific random number of minutes before a job starts; can be used to

ensure jobs are not started on top of each other

Configure a System to Use Time Services

- **timedatectl list-timezones**
  - Command will list all available time zones
- **tzselect**
  - A tool that helps the user select the appropriate time zone. It will then display the time zone setting name to be used. This command DOES NOT make changes to the system.
- **timedatectl set-timezone America/Chicago**
  - Set the system time zone
- **timedatectl set-time**
  - Set the system time
  - **timedatectl set-time YYYY-MM-DD hh:mm:ss**
    - the date or time can be omitted when using this command
  - **timedatectl set-time 14:00:00**
    - Set the systemtime to 2PM
- **timedatectl set-ntp true|false**
  - Enable the usage of the NTP time protocol (Network Time Protocol)
- NTP Configuration: The NTP (Network Time Protocol) can be managed by the chronyd service or the ntpd service. Chronyd is more used for systems that frequently turn off such as portable devices. However, it is an acceptable solution for always on servers as well. Generally, ntpd is used for always on servers.
- Chronyd is the default user space deamon in Red Hat enterprise Linux 7, ntpd can be used but only if chronyd is disabled.
- iburst - the default when configuring servers in the chrony.conf file. It will send a group of 8 packets instead of one at every poll interval. This reduces the time taken for initial syncrhonization.
- chronyc sources -v
- **systemctl restart chronyd**

  - m -> m column represents the state of the sources
    - * indicates the source to which the chronyd service is currently synchronized
    - + indicates acceptable sources
    - ? indicates a source that connectivity has been lost from or whose packets are not passing tests
    - x represents a clock that chronyd thinks is “false” which means it is not consistent in its time checking job
    - ~ a source which appears to have to much variableid
- chronyd vs ntp and when which would be used
- crhonyc tracking
  - Check to see if the chrony time is synchronized

Install and Update Software Packages from Red Hat Network, a Remote Repository, or From a Local File System

Yum is a command line package management tool that allows the querying, downloading, and updating of software packages available from a remote repository. Yum, at its core, will download RPM packages available from the repository and all the required dependencies. When removing yum packages, you cannot remove them without having the package dependencies removed when the package was originally installed.

Yum commands:

- **yum help**
- **yum install package name**
- **yum search string**
  - The string will search the package details and description for anything that matches the "string". You can also pass basic regular expressions in the search string. By default, this will search package names and summaries. However, if it fails, it will also try descriptions and URLs.
- **yum search all**
  - Search all packages that have "string" in their name, summary, and description fields.
  - By default, searches descriptions and URLs as well as package name and summaries.
- **yum list**
  - Will display all installed packages currently on the system
- **yum list all**
  - Will list all installed AND available packages on the systemv
- **yum list installed**
  - Will list installed packages
- **yum list installed httpd**
  - Will list installed packages and search for the httpd package
- **yum check-update**
  - Will list all packages on the system that have an update available
- **yum update package_name**
  - Will update just a specific package
- **yum update**
  - Will update all packages and their dependencies; on a Red Hat system, yum upgrade and yum update perform the same function.
- **yum info httpd**
  - Provide information about httpd package
  - If the package is not installed under "repo", it will display the repo it is currently pulling from.
  - If the package is installed, it will under repo show "installed".
- **yum provides /var/www.html**
  - yum provides will display packages that match the pathname specified
- **yum list kernel**
  - List all installed and available kernels; uname -r and uname -a will show kernel version and release information.

- **yum remove package_name**
  - Remove a given package from the system. This command will remove the given package AND any packages that REQUIRE this package. Be careful as it can result in unexpected removal of packages and could potentially cause issues on the system.

Working with yum groups
Yum groups are a “grouping” of software packages that are used to install the state of a system. For example, yum group install "Server with GUI” will install all the packages required for the system to run with server and a GUI based interface.

- **yum grouplist**
  - Show available groups to be installed
- **yum grouplist hidden**
  - Will show all available groups.
- **yum grouplist ids**
  - Will show the available groups that can be installed as well as the package-id that can be used to install the package instead of the group name.
- **yum groupinstall “group name"**
  - Use quotes to install a group name since they have spaces. Alternatively, you can also use the group ID if it is available.
- **yum groupinfo “group name"**
  - Will display all the packages that will be installed with the group
  - **“-“** - Means the package is not installed and WILL NOT be installed if the group is installed or updated
  - **“=“** - Means the package is installed and was installed as part of the group
  - “+” - Means the package is not installed but it will be installed on the next yum upgrade or yum group upgrade
  - “no symbol” - Package is installed but was not installed by the package group and was installed separately. This means that the command yum group remove WILL NOT remove the package.

YUM transaction history
Whenever software is installed or removed using yum, a history of each transaction is kept in the yum program. One of the key advantages of this is that you can “roll back” specific transactions if needed.

- **/var/log/yum.log**
  - Location of the yum log file
- **yum history**
  - Provides a summary of all install and remove transactions
- **yum history undo “id number"**
  - Undo a specific transaction located in the history. The id number is taken from the column on the row for the transaction that you wish to roll back.

Enabling Third Party Repositories
Note: repositories located in /etc/yum.repos.d must end in .repo in order for them to be recognized.

- **yum repolist**

    - List the repository ID, name, and number of packages that are available for each **enabled** repository on the system
- **yum repolist -v**
    - List more information about the repositories
- **yum repoinfo**
    - Show information about all enabled and disabled repositories
- **yum repolist all**
    - List all repositories active and not active
- /ec/yum.repos.d/Red Hat.repo
- **yum-config-manager**
    - **yum-config-manager** --enable reponame
    - **yum-config-manager** --disable reponame
    - **yum-config-manager** --add-repo repourl

Working with RPM

Packages are created into RPM packages for install. Yum's job is to actually download the RPM package and all required dependencies on the system. It is best practice, when possible, to always use yum to download and install packages. This enables better dependency, update, and general package management for packages on your system. However, we will learn how to use the RPM command to install packages. We will also learn how to use the RPM command to list package information like files installed as part of the package and documentation installed with the package.

- RPM

i	install
v	verbose
e	remove a package
h	print 50 hash marks for progress, use with -v for a clean display
U	upgrades or installs a new package
-F	upgrade a package if it is already installed
-q	query a package
-a	all packages
qa	print all installed packages
ql	list files in an installed package
qd	list documentation of an installed package
qpl	list all files in an RPM package

- **rpm -ivh package.rpm**
    - install package.rpm
- **rpm -qa | grep package**
    - query to see if package is installed
- **rpm -ql package**
    - query all files installed as part of the package
- **rpm -qd package**
    - query all files that are documentation as part of the package
- **yum localinstall (This is legacy usage only)**

    - installs the package using yum
    - **yum install file.rpm** is how it should be used now

Create, Delete, and Modify Local User Accounts

- id
- UID ranges for Red Hat users
    - 0 is root
    - 1-200 are "system users" for Red Hat processes
    - 201-999 are system users used by system processes that do not own files on the file system
    - 1000+ are regular assignment to users
- ps au (a will show option to view all processes with a terminal; to see the user associated with the process press u)
- /etc/passwd this is user login/password information
- /etc/shadow the password hash for the user
    - username:password:uid:gid:gecos:/home/dir:shell
    - uid   - the users id
    - guid   - the user's primary group id number
    - gecos is arbitrary text which is something like the users first and last name
    - home dir is the users home directory
    - shell is the shell the user will use when logged in
- Primary group
    - Each user can have exactly one primary group. What does this mean? When a user creates a new user or file, the group owner of that file will be that of the primary group.
- Supplementary groups
    - Each user can belong to several supplementary groups to be granted access to those permissions.
- /etc/groups is the directory that lists what groups a member belongs too
    - groupname:password:GID: list of users comma separated in the group
- getent group username
    - Queries the /etc/groups file and returns the entry for the specific username

Creating users

- useradd --help

Modifying users

- usermod --help

Deleting users

- userdel
    - -r will remove the users home directory

Change Passwords and Adjust Password Aging for Local User Accounts

- /etc/shadow
    - name:password:lastchange:minage:maxage:warning:inactive:expire:blank
    - name - the users login name, must be a valid account on the system
    - encrypted password is the password field; when there is an "!" in front, it means the account is locked
    - lastpassword - date of the last password change
    - minimumage - minimum number of days before a password may be change; a 0 means that there is no minimum
    - maxage - maximum number of days before a password MUST be changed
    - warning - means that a password is about to expire
    - inactive - the number of days an account remains active after a password has expired (allows you to login and change the password); after the specified number of days, the account is automatically locked
    - expiration date - number of days since 1970-01-01 (unix epoch)
- usermod -s /sbin/nologin
    - allows the user access to the system but not an interactive shell (login via the terminal)
- usermod -e (number of days since 1970-01-01)
    - usermod -e 1 anthony -> would lock the user anthony immediately

Chage is a command used to modify the number of days between password changes and the date of the last password change. It is used to enforce password policy.

- -d Set the number of days since 1970-01-01 when the password was last changed. You can also specify the "date" yyyy-mm-dd.
- -E Set the expire-date, number of days since 1970 OR yyyy-mm-dd to expire.
    - passing -1 as the expire date will remove an account expiration date.
- --help
- -I (uppercase i) Set the number of days of inactivity after a password has expired before the account is locked.
- -l Show account aging information
- -m Set the minimum number of days between password changes; 0 indicates that user may change the password at anytime.
- -M Set the maximum number of days during which a password is valid (until it needs to be changed); passing -1 will remove checking the passwords validity.
- -W Set the number of days of warning before a password change is required.
- date -d '+90days' +%F

Create, Delete, and Modify Local Groups and Group Memberships

- groupadd
    - -g (group id)
    - -r Create a system group
- groupmod
    - -g specify a new group id

    - -n   specify a new group name
- groupdel groupname
- usermod -g primary_group_name username
    - Changes the users primary group
- usermod -G
    - Changes the users supplementary group. If not passed with the -a option, it will replace all existing supplementary groups with just this one.
- usermod -aG
    - "Appends" the additional supplementary group to the user; this way the user can belong to multiple groups at one time.

Set-GID on directories
Users on a system that belong to multiple groups or need to share and edit multiple files might be creating files in specific directories. In order to ensure the correct file ownership inside of a directory, the set-GID permission bit is available. For example, a user's primary group and they belong to a supplementary group called group2. When the user creates a file in group1 directory, the ownership of the file they create will be that of the primary group. Instead of the user having to use the newgrp command to log in and out of groups, you can modify a groups set-GID bit. When set, then all files/directories created inside of a specific folder will automatically have permissions of the group that owns the parent directory instead of the primary group of the local user on the system. The Set-GID bit has two behaviors - one, if you are executing inside of a directory and two, if you are executing a file.

- **chmod g+s dirname**
    - When set, the file executes as the group that owns the file.
    - When set on a group, files that are created in the directory will have permissions set to match the group owner of the directory.

Create, Mount, Unmount, and Use vfat, ext4, and xfs File Systems

vfat: vfat is an extension of the fat file system. vfat supports long file names and is often used when creating shares between a Linux machine and Windows machine. This is common in environments when using services like samba or sharing files on USB sticks between a Windows and Unix machine.

- **Create the vfat file system on a device**
    - mkfs.vfat /dev/xvdf1
- **Mount the vfat file system and device**
    - mount /dev/xvdf1 /mnt/location
- **Add to /etc/fstab for automounting on startup**
    - Mount in fstab normally; example, /dev/xvdf1 /mnt/location vfat defaults 1 2
- **Check file system for consistency (only if the file system is not mounted)**
    - fsck.vfat /dev/xvdf1

Ext4 files system: Ext4 file system is commonly used among Linux systems because of its previous version ext2 and ext3. Ext4 is a journaling based file system and can support file size on Red Hat 7 of up to 16Tb in size and up to a 50TB file system size.

- **Create the ext4 file system on a device**
  - mkfs.ext4 /dev/xvdf1
- **Mount the vfat file system and device**
  - mount /dev/xvdf1 /mnt/location
- **Add to /etc/fstab for automounting on startup**
  - Mount in fstab normally; example, /dev/xvdf1 /mnt/location ext4 defaults 1 2
- **Check file system for consistency (only if the file system is not mounted)**
  - fsck /dev/xvdf1
- **Get details about the file system**
  - dumpe2fs /dev/xvdf1
- **Label the device**
  - tune2fs -L labelname /dev/xvdf1

XFS file system: XFS file system is known for its paralleling processing and high i/o throughput. It is a journaled file system and supports up to 500TB in file size on Red Hat 7 and 500TB in files system size on Red hat7.

- **Create the xfs file system on a device**
  - mkfs.xfs /dev/xvdf1
- **Mount the vfat file system and device**
  - mount /dev/xvdf1 /mnt/location
- **Add to /etc/fstab for automounting on startup**
  - Mount in fstab normally; example, /dev/xvdf1 /mnt/location xfs defaults 1 2
- **Check file system for consistency (only if the file system is not mounted)**
  - xfs_repair /dev/xvdf1
- **Get details about the file system**
  - xfs_info /dev/xvdf1
- **Label the device**
  - xfs_admin -L labelname /dev/xdf1

Note: All other objectives can be found in the "Advanced Topics Study Guide". The study guide is separated on harder concepts for more focused studying.