RHEL7 John Bryce Course

Contents

[RH124 COURSE](#h.pcysl5eb4gkn)

[The Basics](#h.p4a3bnepvuoq)

[Commands](#h.vf35cgm6g4vp)

[Files](#h.9pc4cx6zef91)

[Files hierarchy](#h.t3hji6otmaz4)

[Paths:](#h.min8jxleod5l)

[Text files creation](#h.1zoeaug9bt20)

[Read files](#h.uvw9bp1b7arl)

[Manage files](#h.qx76pzgk08q3)

[Wildcards](#h.up902o3k59mt)

[Search the local documentation](#h.xzep0vfef3wy)

[I\O Redirection](#h.ab5tktzeg6tz)

[| Pipes](#h.ns4jxqjimrke)

[vi(m) text editor](#h.obethdz5it2o)

[Users & Groups](#h.65010frlslpw)

[Permissions](#h.mv5vww8hb5ex)

[Special Permissions](#h.w9vbrv2hqpph)

[Processes](#h.vh18w27wct78)

[Services](#h.dste7i3bi35y)

[SSH](#h.5o0gd16kjovr)

[Logs](#h.yxnanrae7g4m)

[chronyd](#h.zfqadhz2e37)

[Network](#h.ol5bst4ynpgf)

[Copy between machines](#h.tc4i5g5hcdbd)

[Installs](#h.kamjkhrw383f)

[compiling](#h.e2udsllx6498)

[rpm](#h.trarejtaldlh)

[yum](#h.grkwwf2sl6dy)

[File systems](#h.9kkwsrrcgefr)

[Block Device](#h.26mexht1cvbb)

[Create file system](#h.qaox7e7krf2b)

[Shell Scripting](#h.f6rf2o4f9r78)

[Wildcards](#h.dyplp69703vf)

[Output Redirection](#h.88gi38p0l36j)

[Input Redirection](#h.x633anf8xoj6)

[Pipes and Filters](#h.jwkxzcu7wpbm)

[Pipe](#h.jrqja8j0cdkq)

[line filters](#h.u8l80k9nsmil)

[Column filters](#h.yf8gcc1mzcow)

[Regular Expressions](#h.kwq0ytjh9v2v)

[Advanced find](#h.ltbptbo534ml)

[Variables](#h.lhz8trg0gmva)

[Scripts](#h.93tau72zq6e4)

[RH134](#h.vcx2xbtxd17m)

[Kickstart](#h.an6cp1hef16n)

[Scheduling](#h.cbsjdz8kfddj)

[Access Control List](#h.8zp0y3q2n9nc)

[SELinux](#h.5px7qf5nr4fc)

[LDAP & IPA](#h.z88994z7h284)

[Block Devices](#h.wha6u6h520q)

[LVM](#h.ydphosb7l69w)

[NFS](#h.jkm2mprsg38h)

[Boot Process](#h.hpe0cuxqtyb7)

[Firewalld](#h.o1dul9577j8d)

[RH254](#h.diso7yyb19ql)

[Systemd](#h.n249ahloyue2)

[Boot process](#h.hyw4tvbzfas2)

[IPv4](#h.n1qrhzym3xy9)

[IPv6](#h.n417gywbx2fa)

[Link Aggregation](#h.cma3wjo71bhq)

[Bridging](#h.xld14iczii9k)

[DNS](#h.mmykdih8i3qy)

[Configuring Email Transmission](#h.b5omlfxpircw)

[Remote Storage](#h.gx0ycgn7iju1)

[NFS](#h.y6ezqnishiid)

[Samba](#h.xzzu31mtbhoz)

[MySQL / MariaDB](#h.deqkmzf6e0wl)

[Apache](#h.uzkx788hz119)

[Bash scripting](#h.ppz5lt2d7naw)

## 

## 

# RH124 COURSE

## The Basics

Kernel is the core set of the OS functions.

Different kernel require different commands.

The kernel is divided to 2:

1. Monolithic - the core of the kernel on which we add:
2. Modules - added functions, e.g. storage, firewall, File Systems etc..

In order to maintain an effective kernel we remove unnecessary modules.

The kernel is surrounded by the shell, which is the UI (GUI or CLI) to the OS.

There are also TUI, menu driver interface, e.g. setup

CLI Shells:

* Bourne Shell
* Korn Shell
* (TC)C Shell
* BASH - supports Bourne and Korn shells
* Z Shell - more advanced than BASH, supports C Shell as well

ctrl+alt+f1-f4 - Move between desktops

Remote connection to OS:

1. telnet - not encrypted, not supporting some encoding
2. rsh - remote shell, not encrypted
3. ssh - secure shell, encrypted, use V2 to support RSA (V1 uses DSA)

Users:

1. root - has full administrative privileges, has permissions which are hard coded. E.g. root doesn’t require permissions to read file.  
   The privileges are derived from the UID 0 assigned to it.
2. rest of the users.

Users can have the same UID (Unlike the Windows SID).

This way can create different user with the same UID and same power.

running only *su* will maintain the same environment as the original user logged in with.

su - acts as interactive login and will start all of root’ environment, including home dir and startup scripts.

su -c will run only one command, e.g. su -c “head -1 /etc/shadow”

su -s allows substitution to root using shell, even if root has no default shell

Display a couple of elements using {}, e.g. ls -l /etc/{passwd,shadow}

After connection has been established 2 things will happen:

1. Open shell
2. Go to home dir

Linux is a case-sensitive system.

## Commands

**Structure** is divided to 3:

command [-options] [arguments]

command is what to do?

[-options] how to execute?

[arguments] on what to execute?

[] indicates optional parameter.

For example: ls -a /boot

ls command lists folder content.

-a option displays hidden files, indicated by files starting with .

/boot argument means the command and its option will be executed on the /boot dir

The difference between using - before an argument and the use of --:

1. Using - will use the short argument version and can concatenate arguments.  
   Example: ls -al will display all files, including hidden (a argument), and their attributes (l argument).
2. Using -- means to only one argument.  
   Example ls --all will display all files, including hidden, same as ls -a only.

Getting help:

1. man COMMAND - will output the manual pages for the COMMAND.  
   Move inside the man command:  
   /SEARCHTERM - search forward  
   ?SEARCHTERM - search backward  
   q - quit  
   G - go to manual start  
   ###G - go to line number ###  
   /pattern \* Search forward for (N-th) matching line.  
   ?pattern \* Search backward for (N-th) matching line.  
   n \* Repeat previous search (for N-th occurrence).  
   N \* Repeat previous search in reverse direction.  
   ESC-n \* Repeat previous search, spanning files.  
   ESC-N \* Repeat previous search, reverse dir. & spanning files.  
   ESC-u Undo (toggle) search highlighting.  
   &pattern \* Display only matching lines  
   man -k SEARCHTERM - search for manual pages with the specific SEARCHTERM, example man -k list  
   whatis COMMAND - return what is the command, taken from the COMMAND manual synopsis  
   whereis COMMAND - command location, taken from the COMMAND manual synopsis  
   man # COMMAND - go to section # in the manual
2. COMMAND --help - will display the COMMAND’s help file.

Man pages are divided into 9 numbered sections:

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 eg /etc/passwd

6 Games

7 Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)

8 System administration commands (usually only for root)

9 Kernel routines [Non standard]

ctrl + u - cleans line from indicator and start of line

ctrl + k - cleans line from indicator and end of line

history - display command history

touch file$(date +%d%m%Y).log - create file with today’s date, e.g. file22112015.log

date formats:

Display date in dd\_mm\_YYYY format: date +%d\_%m\_%Y

Output: 22\_11\_2015

Display seconds since 1/1/1970 for last friday 00:00: date +%s -d "last friday"

Output: 1447970400

Display seconds since 1/1/1970 for 2 days ago: date +%s -d "2 days ago"

Output: 1448046035

Basic commands:

file - retrieve file information.

Example: file /etc/passwd

Output: /etc/passwd: ASCII text

head - display # of lines since file start, default 10

Example: head -n 3 /etc/passwd

tail - display # of lines since file end, default 10

Example: tail -n 3 /etc/passwd

To run tail in a loop, e.g. monitor log file, use -f. Example: tail -f MonitoredFileName

watch is used to monitor changes in command output every 2 seconds. For example, to view the contents of /var/log/sa execute: watch ls -l /var/log/sa

wc - word count, -l argument will provide only line count

Examples:

wc -l /etc/passwd

Output: 42 /etc/passwd

42 is the number of lines

wc /etc/passwd

42 86 2167 /etc/passwd

42 lines count

86 word count

2167 bytes

history - commands history, default last 1000 commands

!! - will rerun last command

!# - will rerun command #. e.g !999 will rerun command #999.  
!-# - will rerun command that was executed # ago

!string - will run the last command that started with string

!$ will return the last argument we worked on

Example: ls -l /etc/hosts

Output:

-rw-r--r--. 1 root root 242 Nov 22 20:05 /etc/hosts

Now use !$ using another command: wc !$

Output:

wc /etc/hosts

5 17 242 /etc/hosts

## Files

Filetypes:

1. common
   1. - indicates **file**, for example:  
      -rw-r--r--. 1 root root 589812 Oct 6 06:26 initrd-plymouth.img
   2. d indicates **directory**, e.g.:  
      drwxr-xr-x. 2 root root 26 Oct 6 07:16 grub
   3. l indicates **soft link**/symbolic link (shortcut), e.g.:  
      lrwxrwxrwx. 1 root root 11 Oct 6 06:22 /etc/init.d -> rc.d/init.d
2. Devices (hardware):
   1. b **Block devices** - storage devices, since they store blocks
   2. c **Character devices** - added devices, named this way since they provide characters, e.g. keyboards, audio cards etc…
3. IPC (Inter Process Communication)
   1. p **Pipe** | - passes output from one process to the other, the pipe devices store the output while passing it.
   2. s **Socket** - temporary stores data, for example when communicating from BASH to local DB, socket files store the BASH commands sent to the DB.

Files are divided into classes (types).  
Linux is case-sensitive so one cannot create 2 files with the same name in the same dir.

E.g. one cannot create a file named ‘file’ and a directory named ‘file’ in the same folder, since both are files:

[amit@clab1 ~]$ touch file

[amit@clab1 ~]$ mkdir file

mkdir: cannot create directory ‘file’: File exists

In Linux the file extension doesn’t mean anything.  
What happens with a file depends on the command that we executed on it.

For example, if there are 2 files with the same content 1.txt and 1.ggg, their cat output will be identical:

[amit@clab1 ~]$ echo hi > e.txt

[amit@clab1 ~]$ echo hi > e.ggg

[amit@clab1 ~]$ cat e.txt

hi

[amit@clab1 ~]$ cat e.ggg

hi

file checks the data structure of the file, so even if its extension will be changed it will remain the same type (compressed, ASCII etc…)

\ in linux is used as escape character.

#### Files hierarchy

**/** root folder, not the folder of the root user, but the root of the system, no files can exist above it.

**/usr** holds 3rd party program files.

**/bin** holds the commands that came with the system other than those that manage it (basic binaries)

**/sbin** holds commands that manage the system (system binaries), for example shutdown

**/local** - Software (tar.gz installations, non RPM)

**/lib** holds library files (shared objects), equivalent to Windows dll

**/lib64** shared libs 64bit

**/bin** softlink to /usr/bin

**/sbin** softlink to /usr/sbin

**/lib** softlink to /usr/lib

**/lib64** softlink to /usr/lib64

**/etc** holds configuration files

**/var** holds files that are not temporary and are changed during system runtime, for example printer spooler and system logs, also hold Data files (www, mysql, ftp)

**/run** holds temporary files that are changed during system runtime and are not maintained across reboots, e.g. disk on key mount points

**/boot** holds the boot files: kernel image (vmlinuz\*), bootloader (grub2)

**/home** default path for user’s home directory. But, one can change the user’s home directory to another location

**/root** root’s home directory

**/dev** holds files that present devices, for example /dev/sdb represents Serial Disk B (/dev/sda is holding the system disk)

**/mnt** holds the mount points for the non-system disk, for example /mnt/media points to /dev/sdb. Can be configured to be done automatically.

**/proc** RAM filesystem, exposes kernel data and configuration, recreated every boot, e.g. /proc/cpuinfo holds cpu related information

**/sys** RAM filesystem, exposes hardware data, recreated every boot

**/tmp** temp files

**/media** dynamic mountpoints, CD-Rom or disk on key

**/opt** 3rd party apps, empty by default

**/** can be combined out of 1 physical disk, but also from many more aggregated together.

#### **Paths**:

* **Absolute** - the full path to a file, **always starts with /**. For example access the file 1.txt under amit’s home directory from everywhere is: /home/amit/1.txt
* **Relative** - the path in relation to the current location, **never starts with /**. For example if located at /home/amit/dir1/dir2, access using ../../1.txt

Can always use either absolute or relative.

In case nothing is mentioned, relative is implied to the current directory, also indicated by a dot .

~ indicates the user’s home directory. It is an absolute path, but it is changed per user.

ls - short for list, main parameters:

h - human readable file size

d - do not display file under the directory, for example:

[amit@clab1 ~]$ ll /boot/

total 175784

-rw-r--r--. 1 root root 123891 Sep 15 15:14 config-3.10.0-229.14.1.el7.x86\_64

-rw-r--r--. 1 root root 123838 Mar 6 2015 config-3.10.0-229.el7.x86\_64

-rw-r--r--. 1 root root 162752 Oct 3 15:14 config-4.2.3-1.el7.elrepo.x86\_64

drwxr-xr-x. 2 root root 26 Oct 6 07:16 grub

drwxr-xr-x. 6 root root 104 Oct 6 07:44 grub2

-rw-r--r--. 1 root root 39856983 Oct 6 06:27 initramfs-0-rescue-8eb2dcce4a834606b5dcff8cc155cc52.img

-rw-------. 1 root root 17614300 Oct 6 07:19 initramfs-3.10.0-229.14.1.el7.x86\_64.img

-rw-------. 1 root root 18266335 Oct 6 07:23 initramfs-3.10.0-229.14.1.el7.x86\_64kdump.img

-rw-------. 1 root root 18116025 Oct 6 06:30 initramfs-3.10.0-229.el7.x86\_64.img

-rw-------. 1 root root 18259501 Oct 6 07:04 initramfs-3.10.0-229.el7.x86\_64kdump.img

-rw-------. 1 root root 17957956 Oct 6 07:36 initramfs-4.2.3-1.el7.elrepo.x86\_64.img

-rw-------. 1 root root 18278007 Oct 6 07:43 initramfs-4.2.3-1.el7.elrepo.x86\_64kdump.img

-rw-r--r--. 1 root root 589812 Oct 6 06:26 initrd-plymouth.img

-rw-r--r--. 1 root root 240326 Sep 15 15:16 symvers-3.10.0-229.14.1.el7.x86\_64.gz

-rw-r--r--. 1 root root 240039 Mar 6 2015 symvers-3.10.0-229.el7.x86\_64.gz

-rw-r--r--. 1 root root 292829 Oct 3 15:16 symvers-4.2.3-1.el7.elrepo.x86\_64.gz

-rw-------. 1 root root 2881776 Sep 15 15:14 System.map-3.10.0-229.14.1.el7.x86\_64

-rw-------. 1 root root 2881257 Mar 6 2015 System.map-3.10.0-229.el7.x86\_64

-rw-------. 1 root root 3360216 Oct 3 15:14 System.map-4.2.3-1.el7.elrepo.x86\_64

-rwxr-xr-x. 1 root root 5029136 Oct 6 06:27 vmlinuz-0-rescue-8eb2dcce4a834606b5dcff8cc155cc52

-rwxr-xr-x. 1 root root 5029232 Sep 15 15:14 vmlinuz-3.10.0-229.14.1.el7.x86\_64

-rwxr-xr-x. 1 root root 5029136 Mar 6 2015 vmlinuz-3.10.0-229.el7.x86\_64

-rwxr-xr-x. 1 root root 5634608 Oct 3 15:14 vmlinuz-4.2.3-1.el7.elrepo.x86\_64

**[amit@clab1 ~]$ ll -d /boot/**

**dr-xr-xr-x. 4 root root 4096 Oct 6 07:43 /boot/**

cd - change directory, main parameters:

cd - - returns back to previous dir in which I was, example:

[amit@clab1 ~]$ pwd

/home/amit

[amit@clab1 ~]$ cd /

[amit@clab1 /]$ pwd

/

[amit@clab1 /]$ cd -

/home/amit

[amit@clab1 ~]$ pwd

/home/amit

cd ~ - go to user’s home dir

Only cd alone also does to user’s home dir

pwd - print working directory

#### Text files creation

touch - creates files. e.g. touch file1 file2 will create files with the names file1 and file2.  
It also touches them and updates their last modified time.

touch file{3,4).log will create file3.log and file4.log

Use of {} can be done evrywhere.

echo - echos the input contents.

> overwrites

>> appends at the end of the file

#### Read files

cat - output file contents to console.

head -# file - show # of lines from file start

tail -# file - show # of lines from file end, view log file in realtime: tail -f log

more - displays file in pages

less - similar to more, has built in search

#### Manage files

cp - copy files, -r means recursive copy, used to copy directories

mv - move files, no need for -r, implicitly performs the operation recursively. Also used to rename a file by moving the file to the same directory using another name

rm [-rif] - delete, -r for directories recursively including their files, -i interactively ask for confirmation, runs implicitly by default, -f forces deletion, cancels -i

mkdir [-p] foldername - creates new directory, -p creates also the parent, for example:

[amit@clab1 ~]$ ll

total 0

drwxrwxr-x. 3 amit amit 17 Nov 25 17:40 dir1

[amit@clab1 ~]$ mkdir -p dir3/dir4

[amit@clab1 ~]$ ll

total 0

drwxrwxr-x. 3 amit amit 17 Nov 25 17:40 dir1

drwxrwxr-x. 3 amit amit 17 Nov 25 17:41 dir3

rmdir - deletes only empty directory

#### Wildcards

|  |  |  |
| --- | --- | --- |
| Character | Meaning | Examples |
| \* | Every character, no matter how long | rm \* will delete all files (other than the hidden ones):  .a  a  a.  a.c  a.log  1.  1.c  1.log |
| ? | Every character that appears only once | rm \*.? will delete:  a.c  1.c |
| [abc\*],[a-z],[0-9a-zA-Z]  Note: MUST write [] | Every character from the list only once, at the target start or end. The list gets its most simplest meaning, for example [abc\*] means a or b or c or \* (not the wildcard, the \* sign specifically) | rm [a-z] will delete:  a.  a.c  a.log |
| [!a-j],[^a-j] | Both ! and ^ exclude characters in the list only once, in the start or end of the list | rm [!a-z].[!0-9]\* will delete:  1.c  1.log |

Use command output inside file creation, for example copy file to a new file with YYYY-mm-dd appended:

cp file1.odf file1\_$(date +%Y-%m-%d).odf

Example:

[amit@clab1 ~]$ touch file1.odf

[amit@clab1 ~]$ cp file1.odf file1\_$(date +%Y-%m-%d).odf

[amit@clab1 ~]$ ls

file1\_2015-11-25.odf file1.odf

$(COMMAND) is used for command substitution, when the output replaces the input.

apropos is a command equivalent to man -k SEARCHTERM, which unlike man also supports wildcards in the SEARCHTERM.

display command examples, for supported commands: man COMMANDNAME examples

For example: man nmcli examples

info and pinfo display the man page including links, pinfo also includes colors.

Since they are different commands they might not display the same output, excluding colors.

u Go Back

n Go next

/usr/share/doc contains user manual pages.

elinks is a tui based HTML viewer for the CLI.

## Search the local documentation

whatis COMMAND or man -f COMMAND are used to display all related man pages to a command.

Example: [amit@localhost ~]$ whatis mkdir

mkdir (1) - make directories

mkdir (1p) - make directories

mkdir (2) - create a directory

mkdir (3p) - make a directory

apropos SEARCHTERM or man -k SEARCHTERM are used to search inside man pages. Example:

amit@localhost ~> apropos postscript

cupstestdsc (1) - test conformance of postscript files

enscript (1) - convert text files to PostScript, HTML, RTF, ANSI, and overstrikes

man -f COMMAND is used to display the intro of all related man pages. Example:

amit@localhost ~> man -a mkdir

--Man-- next: mkdir(1p) [ view (return) | skip (Ctrl-D) | quit (Ctrl-C) ]

--Man-- next: mkdir(2) [ view (return) | skip (Ctrl-D) | quit (Ctrl-C) ]

--Man-- next: mkdir(3p) [ view (return) | skip (Ctrl-D) | quit (Ctrl-C) ]

Rebuild man search database, use if failing to find: man -u man

If exists, examples in a man page will appear under the EXAMPLES section.

## I\O Redirection

E.g. echo A -> filename

No output on screen, will be written to filename

In case we output to file a non-existent output - we will get an error on the screen.

For example, if we run command on non-existent file - get error:

[amit@clab1 ~]$ ll ggg.fff > 1.txt

ls: cannot access ggg.fff: No such file or directory

0 STDIN - Standard Input (Default: Keyboard)

1 STDOUT - Standard Output (Default: Terminal Screen)

2 STDERR - Standard Error (Default: Terminal Screen)

The rest of the outputs do not have a specific number.

Using the numbers we can direct the outputs to different places.

Example on non-existent file - error will now be written to errors.txt:

[amit@clab1 ~]$ ll ggg.fff 1> 1.txt 2> errors.txt

[amit@clab1 ~]$ cat errors.txt

ls: cannot access ggg.fff: No such file or directory

Errors will always be presented before STDOUT, so in case both are written to the same file, STDOUT will overwrite STDERROR.

For example when file exists and NE doesn’t, so the output about file will overwrite the output about NE:

[amit@clab1 ~]$ ll file NE 1>log 2>log

[amit@clab1 ~]$ cat log

-rw-rw-r--. 1 amit amit 0 Dec 2 15:48 file

In order to output all information will use &> to keep both.

&> equals to 1>> and 2>

Example:

[amit@clab1 ~]$ ll file NE 1>>log 2>log

Equals to:

[amit@clab1 ~]$ ll file NE &>log

In order to drop all output command - redirect the output to /dev/null, which is an always empty place.

E.g. output everything to &>/dev/null.

#### | Pipes

Used to send STDOUT (by default will not send STDERR) from one command to another, by changing the STDOUT to be the STDIN of the command after |

Example, count lines by:

ls > file

and then count using

wc -l file

Using | we can run one command and avoid using the file:

ls | wc -l

In order to send STDERR as well use |&

For example: ls |& wc -l

tee is used to multiply the output (same as T junction). It takes the output, copies it to a file and sends it down along the next output (console screen or pipe).

For example:

[amit@clab1 ~]$ ls | tee FILENAME

file

file1

file10

file11

Pipes can be used multiple times in the same command, example:

ls | tee FILENAME | wc -l

Multiple records in an output are separated using a new line \n by default, example:

[amit@clab1 ~]$ ls > file

[amit@clab1 ~]$ cat file

file

file1

file2

file3

file4

file5

$(CMD) changes the separation to spaces, which allows to work with other commands.

rm requires spaces between the files it is requested to delete, so using command substitution is required. e.g. delete all files in current directory: rm $(ls)

Summary difference between pipe and command substitution:

|  |  |  |
| --- | --- | --- |
|  | **Creates...** | **What does it do with record separator** |
| **Pipe |** | STDIN for next command | Nothing |
| **Command Substitution** | argument for current command | Switches from line separator \n to space separator |

## vi(m) text editor

3 work modes:

1. Insert
2. Command - default mode, the mode vi opens in.
3. Extend

Move to insert mode using:

a - moves the cursor one character and changes to insert mode.

A - moves the cursor to end of line and changes to insert mode.

i - changes to insert mode from current cursor location.

I - moves cursor to start of line and changes to insert mode.

o - opens a new line below the cursor and changes to insert mode.

O - opens a new line above the cursor and changes to insert mode.

Return to Command Mode: ESC

Navigate (command mode):

#G - move the cursor to line #

G - move the cursor to last line

H - move the cursor to first line on screen (not necessarily the first line of the file)

/SEARCHTERM - search downwards for SEARCHTERM

?SEARCHTERM - search upwards for SEARCHTERM

n - move to next result

N - move to previous result

^ - move to current line start

$ - move to current line end

b - move to current word start

e - move to current word end

Edit (command mode):

x - delete character

rX - replace the character at which the cursor is located with the input character

u - undo, can undo until last save

ctrl + r - redo (un-undo), equivalent to ctrl+y in Windows

#yy - copy # of lines below current cursor location, only yy will copy the current line

#dd - cut # of lines below current cursor location, only dd will cut the current line

yy#p - copy the current line the cursor is located and paste it # of times. For example, copy the current line and paste it 15 times execute: yy15p

p - paste under cursor location

P - paste above cursor location

Exit and save (command mode):

ZZ - save and exit

: - change to Extended Mode

:# - move to line #

:set number - display lines number

:set no number - cancel lines number display

Enter - change back to Command Mode

:3,10s/SEARCHTERM/REPLACETERM/[gi] - search between lines 3-10 and substitute SEARCHTERM with REPLACETERM. [gi] is optional, and is not required, tells whether the replacement is global or not.

:%s/SEARCHTERM/REPLACETERM/[gi] - % tells to search in all file and s to search and replace

:!COMMAND - run external command, for example to get a reminder directory location execute :!pwd

:w - save

:w! - force save, if the file is already open elsewhere

:w FILENAME - save file to FILENAME, while remaining to work on current file (unlike Microsoft Save As which changes to write in the new file)

:q - quit without saving, will verify if all changes were saved

:q! - quit without prompt for saving

:wq or :x - save and quit

:wq! or :x! - force save and quit

Visual Block - allows work with mouse

ctrl+v - changes to Visual block, use only the arrow keys to select blocks, using a mouse will select lines

alt+v or v - change to visual mode, per line and not per block

In GUI:

ctrl+shift+c - copy

ctrl+shift+v - paste

If need to open a new bash window and continue to use current console as well, use & at the end of the command which opens the new console window, e.g. gedit FILENAME & will open FILENAME with gedit and at the console will return to prompt instead of wait for gedit session to complete.

## Users & Groups

User - parameters in the /etc/passwd:

name:pass:UID:GID:comment:HomeFolder:Shell

Only the username must be unique.

Password actually saved in /etc/shadow

UID indicated User ID, 2 users with the same UID have the same permissions.

GID is the primary Group ID of the user. There are private groups, with only one user. Common groups contain multiple users which enjoy the same permissions applied for the common group.

Can cancel the use of private groups, so new users will be added to a group named *Users*, as its primary group.

When a user creates a file, he adds permissions to his primary group by default.

Non-primary groups (AKA supplementary groups) provide additional permissions for the user.

Unlike windows, membership in the *root* group, doesn’t provide the same permissions as the root user, only having the same UID as root.

Groups are managed in /etc/group:

name:x(one contained the group password):GID:members

Group passwords are saved in /etc/gshadow

The file /etc/login.defs contains the default user creation policy.

When working with private groups the number sequences of UID and GID are promoted together. In order to maintain the sequence, specifically mention the GID for manually created new groups.

/etc/default/useradd contains the default policy for the useradd command.  
It is recommended to configure both /etc/login.defs and /etc/default/useradd with the same policy.

Commands to manage users:

useradd - apply on /etc/passwd. Important flags

-u - set UID

-p - specify password in hash encryption (create another user with the required password and take the hash from /etc/shadow )

/etc/skel - default skeleton folder, contains the default files in user’s home directory

-mk - use a different skeleton folder

usermod - apply on /etc/passwd

userdel - apply on /etc/passwd

passwd - apply on /etc/shadow

In /etc/shadow the file structure is:

username:PasswordHASH:date of last password change:minimum password age:maximum password age:password warning period:password inactivity period:account expiration date

chage - change password retention etc…, apply on /etc/shadow

id USERNAME - list all the ID’s for a given user

usermod -G “” USERNAME - remove USERNAME from all supplementary group, leaving it a member only of its primary group

Commands to manage groups:

groupadd - apply on /etc/group

groupmod - apply on /etc/group  
-g runs on primary group

-G on supplementary groups (default)

-aG add to another supplementary group (without -a will replace the supplementary groups for the user, instead of adding an additional one)

groupdel - apply on /etc/group

gpasswd - apply on /etc/gshadow (not used anymore). Allows adding or deleting a couple of users into a group in one command.

sudo - means users can run some commands after authenticated specifically.

members of sudoers group managed in /etc/sudoers

e.g. members of the group wheel can run any command as any other user.

Default permissions for sudoer allows to run any command as any other user:

## Allows people in group wheel to run all commands

%wheel ALL=(ALL) ALL

Can be configured specifically for specific groups.

%GROUPNAME/Username SERVERNAME=(Impersonate As) Allowed Commands

Example if want to configure specific user specific commands for specific server:

# Configure WS servers group

WS=WebServer1,WebServer2

# Configure permissions on WS for amit

amit WS=(root) service

So amit can run the command service as root on any server in the WS list.

id is used to display all ID’s for a given user:

[amit@clab1 ~]$ id amit

uid=1000(amit) gid=1000(amit) groups=1000(amit),10(wheel)

## Permissions

-rw-rw-r--. 1 amit amit 5 Dec 6 15:35 filename

- file type, main types: - file d directory s softlink

rw-rw-r--. permissions field

1 number of pointers, or file names. If the file is a hardlink, the numbers will increment, if the file is a softlink it will remain 1. When deleting file - deleting its hardlink, not its real content.

amit file owner

amit group owner, the primary group of the file owner

5 file size in bytes

Dec 6 15:35 last modified date for the file’s metadata

filename indicates filename

In Linux permissions are set on files.

There are 3 instances that can get permissions on file:

* The first 3 letters are about the owning *user* permissions.
* The middle 3 letters indicate the primary *group* owner permissions.
* The last 3 letters indicate the permissions for *other* (all those not in the user and group).

The last . indicate the special permissions status. ‘.’ indicate no special permissions, ‘+’ indicates an ACL entry exists. .

(r)Read - read file and list folder contents.

(w)Write - modify file and folder contents (e.g. add/delete files)

(x)Execute - execute file or access folder

A user has full permissions on his home directory, so root can put a write-protected file there, to which the user can’t write, but can delete as he has permissions on the folder.

chmod used to change permissions.

can set on: u (user) g (group) o (other) a (all)

Symbols method, can be used to replace or add/remove permissions:

‘+’ add permissions

‘-’ remove permissions

‘=’ apply a new set of permissions

rwx type of permissions to set

filename to set permissions on

Example: chmod u=rwx,g=rx,o=r filename

filename permissions output: rwxr-xr--

Example: chmod g+w,o-r filename

filename permissions output: rwxrwx---

Example: chmod = filename

filename permissions output: ---------

Default instance is all and default permissions is no permissions, so all permissions will be denied.

Still, the owner and the root user can modify the filename’s permissions, or another g/o that have permissions from the folder.

Example: chmod +x filename

filename permissions output: --x--x--x

Example: chmod +w filename

filename permissions output: -wx-wx--x

Although if no instance is mentioned then all is implicitly referred to - other are not granted with write permissions in the last example to protect filename.

Numbers method - replaces, doesn’t add permissions:

4 r

2 w

1 x

Example: chmod [0-7][0-7][0-7] filename

[0-7] user

[0-7] group

[0-7] other

Example: chmod 754 filename

filename permissions output: rwxr-xr--

When not mentioned specifically, 0 is implied.

Example: chmod 0 filename

Implied: chmod 000 filename

filename permissions output: ---------

Example: chmod 7 filename

Example: chmod 007 filename

filename permissions output: ------rwx

Default permissions are set based on the application used to create the file.

umask is used to set default permissions per current session.

umask is omitting permissions using numerical values.

The default set is 777.

If umask will be executed with 27, then 027 will be implied and the result will be 750 set by default:

777

-

027

-----

750

In order to save umask set across sessions, change the ~./bashrc or ~./bash\_profile (the files that configure the user environment).

In order to change the settings for all users edit the files /etc/bashrc and /etc/profile

If umask is not configured, then files will be created with 666 by default and directories with 777. The default umask is 002, so the effective permissions are 664 for files and 775 for directories.

Umask comes in 4 digits, e.g. 0022, the first for the special permissions.

Display current umask using umask

4 for SUID, 2 for SGID and 1 for Sticky bit.

stat FILENAME is used to display the special permissions on a file.

Ownership change can be done only by root.

chown username filename - Change file owner

Example: chown amit file1

chown username:group filename - Change file owner and group

Example: chown amit:students file1

chown :group filename (equivalent to chgrp group filename)- Change file group

Example: chown :students file1

In order to delete/create/rename a file, user must have write permissions on the folder.

Execute permissions on a directory allows to browse to it, and Read allows to list its contents.

#### Special Permissions

|  |  |  |  |
| --- | --- | --- | --- |
| Permission | Command | file | folder |
| SUID | u+s  Example output on passwd:  -rwsr-xr-x. 1 root root 27832 Jun 10 2014 /usr/bin/passwd | every person running the file, will execute it with the UID of the file owner.  For example: passwd is executed with root’s UID, which allows updating the user’s password to /etc/shadow | None |
| SGID | g+s  Example:  [amit@clab1 ~]$ mkdir dir  [amit@clab1 ~]$ chmod g+s dir/  [amit@clab1 ~]$ cd dir/  [amit@clab1 dir]$ touch file1  [amit@clab1 dir]$ ll file1  -rw-rw-r--. 1 amit amit 0 Dec 6 17:19 file1  [amit@clab1 dir]$ su  Password:  [root@clab1 dir]# touch file2  [root@clab1 dir]# ll file2  -rw-r--r--. 1 root **amit** 0 Dec 6 17:20 file2 | every person running the file, will execute it with the GID of the owning group. | Every new folder created in the folder, will be created with the owning group of the folder and not the user creating the file. E.g. The folder ‘IT’ has SGID set on it. Files created by Odie will have the IT team as owning group and not Odie’s. |
| Sticky Bit | o+t  Example: [root@clab1 dir]# ll -d /tmp/  drwxrwxrwt. 7 root root 4096 Dec 6 17:20 /tmp/ | None | Only an owner can delete his files. E.g. used in case of files in /tmp, to which everyone write, but we want to avoid everyone deleting files for each other. |

Numerical values:

4 Suid

2 Guid

1 Sticky Bit

This is the first letter in numerical method, by default it is 0.

Example:

[root@clab1 dir]# ll file1

-rw-rw-r--. 1 amit amit 0 Dec 6 17:19 file1

[root@clab1 dir]# chmod 4664 file1

[root@clab1 dir]# ll file1

-rwSrw-r--. 1 amit amit 0 Dec 6 17:19 file1

Upper case S/T means that it doesn’t hide x behind it.

Lower case, means x is also allowed.

## Processes

Software can run all the time or ad-hoc, and is executed using a process. Software executes at least 1 process or more.

Service = Daemon, it runs all the time and executes processes.

Processes execute threads (sub-processes).

top - task manager equivalent.

load average displays how many processes waited for the CPU in the last 5, 10, 15 minutes

can be more than 100% to allow understanding of what is actually required.

example: load average: 0.00, 0.01, 0.0

To toggle the display of the the CPU load per core between aggregated to specific - click 1 while top is running.

free -m is used to display RAM usage.

jobs display running tasks summary.

stopped represents suspended processes.

zombie are processes that were shut but still up.

Tasks: 89 total, 2 running, 87 sleeping, 0 stopped, 0 zombie

%Cpu(s): 0.0 us, 0.1 sy, 0.0 ni, 99.5 id, 0.4 wa, 0.0 hi, 0.0 si, 0.0 st

us %CPU used by user

sy %CPU used by system

ni %CPU used for prioritized process

KiB Mem indicates resident / physical memory

KiB Mem : 1016228 total, 788456 free, 96804 used, 130968 buff/cache

KiB Swap indicated swap memory used.

KiB Swap: 946172 total, 946172 free, 0 used. 782496 avail Mem

Virtual memory in Linux means all memory (physical + swap).

PID process ID per current process session.

PR priority as decided by the operating system. It cannot be changed, only impacted by the NICE values.

NI NICE values allows the user to set priority per process between -20 to 19.

NICE value of 0 is the default value.

Process with lower value - gets a higher priority (this is why it is not nice, because it takes resources from others).

VIRT how much total RAM is used from all RAM types (Resident/Physical, Swap, Null).

Null ram is where pages removed from the RAM are going.

RES how much of the VIRT is in the resident/physical RAM.

SHR how much of the processes RAM is used by others as well.

S process state: R running, S suspended, T stopped and Z zombie.

TIME+ how much CPU time the process received.

top commands:

h - help

f - displayed fields, for example add Parent PID. Choose using space.

Shift+< or Shift+> - change sorting fields

Change nice value: r -> enter PID -> enter value

k is used to kill processes: k -> PID -> signal: 15 (gracefully) or 9 (non-gracefully).

15 (terminate) means stop the process gracefully.

9 (kill) means stop the process now.

nice is a command to execute a process with a predefined nice value.

ps displays processes information per current session.

Just executing ps will show the processes running in my current terminal (for example pts/0 is pseudo-terminal session 0, the first terminal session) with the effective UID (the currently logged on user).

ps aux (a - all processes, u - user oriented sorting, x - display processes not running through a terminal session, hose with ? in the TTY column) displays all processes.

ps -ef (e - same as a, f - additional information) is used to display the parent PID.

ps -l ( l - long format) display priority and nice values.

sleep waits according to the specific number of seconds.

ctrl+z interrupts running foreground process and will suspend it.

ctrl+c terminates the session completely.

jobs displays processes that were manipulated by the user, for example ctrl+z was executed on them or processes running in the background.

fg job# will return the job to run in the foreground. When a job is running in the foreground, I cannot run any other process until it is completed.

bg job# will return the job to run in the background, so i can return to run processes.

Examples:

[amit@clab1 ~]$ sleep 1000

^Z

[1]+ Stopped sleep 1000

[amit@clab1 ~]$ jobs

[1]+ Stopped sleep 1000

[amit@clab1 ~]$ fg 1

sleep 1000

^Z

[1]+ Stopped sleep 1000

[amit@clab1 ~]$ jobs

[1]+ Stopped sleep 1000

[amit@clab1 ~]$ bg 1

[1]+ sleep 1000 &

COMMAND that ends with & will be executed in the background.

Start nice with a nice value: nice -n NiceValue command

Priority values range between -99 to 40.

Users can set a nice value between 0-19.

root and users with the required permission can also set a nice value between -20 to 0.

The lower the value - the higher priority assigned to it.

If creating a process with a nice value, it and its sub-processes, will continue to run with this value.

E.g., in order to run sleep of 60 seconds with a nice value of - 20 and display its nice value:

[root@clab1 ~]# nice -n -20 sleep 60 &

[1] 2429

[root@clab1 ~]# ps -l

F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD

4 S 0 2405 2319 0 80 0 - 45017 wait pts/0 00:00:00 su

4 S 0 2409 2405 0 80 0 - 28840 wait pts/0 00:00:00 bash

**4 S 0 2429 2409 0 60 -20 - 26975 hrtime pts/0 00:00:00 sleep**

0 R 0 2430 2409 0 80 0 - 30321 - pts/0 00:00:00 ps

renice allows to change the nice value during runtime: renice -n NiceValue PID

Nice values can be useful when one wants to run jobs with lower priority, e.g. indexing.

nohup is used to execute commands while ignoring hup signals (e.g. timeouts).

This means that it will run the command but not using the current terminal session and write the command output to a log file named nohup.out. This will allow the command to continue to run, even if the terminal session will terminate.

Structure: nohup COMMAND

Example:

[root@clab1 ~]# nohup ls -lR / &

[1] 2442

[root@clab1 ~]# nohup: ignoring input and appending output to ‘nohup.out’

[root@clab1 ~]# cat nohup.out | head

/:

total 32

lrwxrwxrwx. 1 root root 7 Oct 6 06:21 bin -> usr/bin

dr-xr-xr-x. 4 root root 4096 Oct 6 07:43 boot

drwxr-xr-x. 21 root root 3180 Dec 9 15:34 dev

drwxr-xr-x. 92 root root 8192 Dec 9 15:34 etc

drwxr-xr-x. 12 root root 4096 Dec 4 15:17 home

lrwxrwxrwx. 1 root root 7 Oct 6 06:21 lib -> usr/lib

lrwxrwxrwx. 1 root root 9 Oct 6 06:21 lib64 -> usr/lib64

drwxr-xr-x. 2 root root 6 Jun 10 2014 media

[1]+ Exit 1 nohup ls -lR /

kill is used to manipulate processes.

Its default signal is 15, which means terminate.

To terminate a running job, using its job number: kill %JobNumber

Example:

[root@clab1 ~]# jobs

[1] Running sleep 1000 &

[2] Running sleep 1000 &

[3] Running sleep 1000 &

[4]- Running sleep 1000 &

[5]+ Running sleep 1000 &

[root@clab1 ~]# kill %2

[root@clab1 ~]# jobs

[1] Running sleep 1000 &

[2] Terminated sleep 1000

[3] Running sleep 1000 &

[4]- Running sleep 1000 &

[5]+ Running sleep 1000 &

jobs -p will display the jobs PID.

Always display the ps aux headers using ;

ps aux | head -1;ps aux | tail

Kill allows to send signals to commands: kill -SIGNAL %JobNumber

Different signals:

-SIGTERM or 15 - terminate running command, the default signal

-SIGKILL or -9 - kills non-gracefully

-SIGSTOP - suspend running command

-SIGCONT - continue running command

-USR1 - dd specific signal, makes dd print I/O statistics during its process

-HUP - reload the service

pgrep is grep for process, allows finding the PID based on the search parameters.

pkill allows searching for the relevant PID and kill it.

killall kills processes based on the search parameter, supports regular expressions, and will kill all the processes it will find.

Run a long command as 1 job using parenthesis, for example in order to write something to file in 1 second interval and run it in the background, execute:

(while true; do echo -n “Whatever” >> OutFile; sleep 1; done)

## Services

pstree displays process parent and child relationship.

In the past the directory softlink /etc/init.d, which points to /etc/rc.d/init.d, contained script files that executed the services.

The problem was that if 1 script got stuck - all following scripts didn’t execute.

The process init was the parent for all processes executed from the traditional /etc/rc.d/init.d.

It had PID 1.

Until RHEL 5, including, init was the method to execute services.

upstart replaced init in RHEL6 but was still called init.

2 main commands:

service - manage services (stop, start, reload etc…)

chkconfig - configure services (start on boot etc..)

init only managed service scripts, which doesn’t allow to create relationships between them.

system.d replaces init and runs everything from /etc/systemd/system/…

Also has PID 1.

system.d manages units: service, mounts and sockets.

This allows to configure relationship between different units.

systemctl is used to manage and configure system.d services.

systemctl will display services list and their status.

systemctl -t service - display only services

systemctl list-unit-files --type=service - display services startup status (enabled - will start automatically, disabled - will not start automatically, static - dependency for another service).

the systemctl mask argument allows hiding a service, by making it a softlink to /dev/null.

## SSH

User open SSH session to remote server

Remote server answers with fingerprint (ssh\_rsh\_key.pub)

fingerprint is written into the ~/.ssh/known\_hosts file

Fingerprint is taken from remote server /etc/ssh/ssh\_host\_rsa\_key.pub

Securing SSH - manage using /etc/ssh/sshd\_config file:

* Disable root access using SSH
* Disable Password Authentication, which is being sent using clear text, and connect using keys. Will also allow remote connection without password authentication:

1. ssh-keygen will create private key (named: id\_rsa) and public key (id\_rsa.pub). Will also prompt for password during creation, which will allow to work using 2 factor authentication (password to key file and a key file).
2. ssh-copy-id USERNAME@SERVERNAME  
   Will ask for password  
   If ssh-copy-id is not available on the specific distro, just copy it to the .ssh/authorized\_keys at destination home dir on the destination server.
3. Will create in the the USERNAME home directory on the REMOTESERVER under ~/.ssh/authorized\_keys
4. Then one can connect remotely by typing ssh USERNAME@REMOTESERVER without being prompted for a password.

-X - allows to open GUI from ssh by executing a command that open a GUI software, for example, to open the GUI using the Kickstart Configurator: system-config-kickstart &

ssh-agent is used to allow use of encrypted certificate without having to reenter the password each time, by loading the private keys to the RAM for the current session.

## 

## 

## Logs

rsyslog (previously syslog and syslogng) is the daemon in charge of writing logs.

All system services, other than the kernel, write their logs to rsyslog.

The kernel writes its logs to systemd-journald (previously written to dmesg) since its messages are written differently and need to be interpreted, and then transferred to rsyslog.

The way messages are referred to syslog are:

facility.priority.content

example:

cron.info.service has started

rsyslog in turn checks to which log file to write the entry based on the RULES inside /etc/rsyslog.conf

rsyslog writes to the relevant file in /var/log/

/var/log/messages is the default log file.

when priority info is mentioned, it means info and all messages with higher priority will be logged to that file, which means all messages.

When the priority none is mentioned it means NOT to log and that there is a specific log file.

Example, all messages will be written to /var/log/messages other than mail, security and cron which will be written to their logs, example provided for cron:

\*.info;mail.none;authpriv.none;cron.none /var/log/messages

cron.\* /var/log/cron

emerg indicates emergency messages, which are written to Console as well.

logger creates log entries. Structure: logger -p facility.priority MessageContent

Example: logger -p cron.emerg HELP!

Should write to the console screen and cron log file.

In order to enhance rsyslog with additional log parsing and and non-default services, add matching configuration under /etc/rsyslog.d

Logs rotation is managed by the process logrotate

Configured under /etc/logrotate.conf

The settings are global for all files under /var/log, and one can add specific rotation policy for one type only.

logwatch is used to create summary of logs per given time.

journalctl is a systemd utility that allows to kernel messages since last boot, since by default it isn’t preserved over reboots, but saves its info to /run

All systemctl info is forwarded to journalctl.

For example, see all messages in verbose mode since last hour for user id 0:

jounalctl -o verbose --since “1 hour ago” \_UID=0

Show all entries from specific PID:

journalctl \_PID=1

In order to maintain jounald logs over boots, configure the dedicated folder /var/log/journal/. Once this folder is created, journald saves its messages there. Procedure:

# Create dir

[root@clab1 ~]# mkdir /var/log/journal

# set permissions on the new dir

[root@clab1 ~]# chown root:systemd-journal /var/log/journal

# Change permissions using SGID

[root@clab1 ~]# chmod 2775 /var/log/journal

# Restart system-journald service

[root@clab1 ~]# systemctl restart systemd-journald.service

Display journalctl entries from the latest and not from the first: journalctl -r

In order to configure all log entries to be written to disk (some are only written to RAM):

1. Edit /etc/systemd/journald.conf
2. Change #Storage=auto to Storage=persistent
3. Reboot

The book contains longer procedure, but one that does not require reboot.

dmesg still exists, but is much harder to read.

## chronyd

NTP is used to sync clocks.

In RHEL7 the daemon in charge of it is chronyd.

timedatectl is the command to manage chronyd

chronyd configuration file is /etc/chrony.conf

stratum weights are used to configure the priority of the NTP server.

The file /var/lib/chrony/drift maintains how much time passed since last sync and forces time change, in case no NTP server is available.

iburst indication at the end of the server name is used during the initial synchronization in case the server isn’t reachable, to send a burst of 8 packets instead of 1:

# Please consider joining the pool (http://www.pool.ntp.org/join.html).

server 0.centos.pool.ntp.org iburst

server 1.centos.pool.ntp.org iburst

server 2.centos.pool.ntp.org iburst

server 3.centos.pool.ntp.org iburst

chronyc is another utility to manage chronyd

To verify current status run: sources -v

timedatectl is used to enable sync using NTP and whether NTP synchronization is successful.

In order to set timezone use, e.g. for Israel: timedatectl set-timezone Asia/Jerusalem

Can also do it using tzselect

## Network

ifconfig -a displays all network related information for wired connections.

iwconfig will display all wireless related connections.

ip is a command to display and manage network related information.

ip addr is the equivalent to ifconfig

ip link will only show linked (connected) connections

ip -s link show DEVNAME show statistical information for linked connections

ip route displays routing information

tracepath URL/IP and traceroute URL/IP will display trace information over UDP

ss provides socket statistics, similar to netstat

nethogs is a recommended network monitoring utility

The network is managed by 2 services:

1. network - the dominant service pre-RHEL7, works against script files from /etc/sysconfig/network-scripts  
   e.g. ifcfg-eth0 contains the configuration for the eth0 card  
   if PEERDNS is configured as no and DHCP provides DNS servers, the DNS configuration will remain as configured manually and will not apply the the DHCP provided DNS.  
   NM\_CONTROLLED means whether the card is controlled by Network Manager.
2. Network Manager - since both network and network manager do the same, it is advised to route network output to /dev/null, since Network Manager is better in RHEL7 (was worse before).  
   Network Manager knows to manage a couple of devices and connections.  
   In network we didn’t distinguish between devices and connections.  
   This way we can create a couple of profiles.  
   Devices represent the physical device.  
   Connections means different configurations: add, show, modify, remove, up and down.  
   When configuring a connection, one can attach it to a device and configure its name, ipv4, gwv4, autostart etc…  
   nmcli is the utility to manage the Network Manager  
   Structure:  
   nmcli OBJECT OPTION  
   Example to configure a connection (AKA Network Profile, connection in RHEL7 is a logical setting, not physical) named static2 on device eth0 from type ethernet:  
   nmcli connection add con-name static2 ifname eth0 type ethernet autoconnect no ip4 10.0.0.1/24 gw4 10.0.0.254  
   Add DNS server on existing connection:  
   nmcli connection modify static2 +ipv4.dns 8.8.8.8  
   Always use + in order to ADD information and not override.  
   In case using Network due to legacy reasons can update its files while working with nmcli by running:  
   nmcli reload

/etc/hosts contains local DNS entries

/etc/resolv.conf contains DNS configuration (will only resolve against the first 3 servers listed), domain suffix and domain search (for bind, was used before DNS and adds domain suffix to resolve).

/etc/sysconfig/network contained hostname in the past

in RHEL7 hostname is controlled in hostnamectl

## Copy between machines

tar is an archiving (not compress) command. It takes a couple of files and archives them to 1 file.

Creation Structure: tar -cv[*g/j/J*]f OUTPUTFILENAME.tar Files\_To\_Archive

c - create archive

v - verbose output

z - compress in gzip

j - compress in bzip2

J - compress in xz which is 7zip

f - file

Extract structure: tar -xv[z/j/J]f FILENAME\_TO\_UNCOMPRESS.tar.[.GZ/XZ/BZ]

x - extract

The extract command will be done to the current directory.

If one wants to extract to specific directory cd to it first and then extract.

List structure: tar -tv[z/j/J]f FILENAME\_TO\_LIST\_TO\_CONTENTS.tar[.GZ/XZ/BZ]

t - list

gz is a compression command. If applied on a couple of files not using tar - will compress each separately.

ftp is a protocol to copy files, port 21.

sftp is the same, just secured using ssh, port 22.

tftp is ftp based on udp.

scp and rsync are non-interactive command that use sftp to copy files.

scp is considered faster than rsync, but rsync knows to sync, which means it will only copy the delta between the files.

scp is most commonly used for ad-hoc copies and rsync for long term files sync.

Examples using scp:

scp [-p|-r] WHAT\_TO\_COPY username@DEST\_Server:WHERE\_TO\_COPY

Or:

scp [-p|-r] username@SOURCE\_Server:WHAT\_TO\_COPY WHERE\_TO\_COPY

a - keep the same attributes (owner, permissions) but **doesn’t** copy recursively

r - copy recursively without the attributes

Copy using rsync:

rsync [-arvzn] WHAT\_TO\_SYNC username@DEST\_Server:WHERE\_TO\_SYNC\_TO

a - keep the same attributes (owner, permissions) **and** copy recursively

r - recursively

v - verbose

z - compress before copying

n - dry run, do not copy but show me what will be copied and how much bandwidth is required.

## Installs

### compiling

5 steps to compile, example on package received through a file named app.tar.gz:

1. Extract install files to current folder:  
   tar -xzvf app.tar.gz
2. Setup by executing the install script:  
   ./configure that will check:
   1. Compatibility check, will tell which files are missing, but not the specific dependency name
   2. Personal customization (note most scripts will come with a help file, display using ./configure --help)

and will create Makefile  
The Makefile has the build, install and cleanup phases configuration.

1. Build by compiling the package (making a binary file out of the source file):  
   Execute make inside the Makefilea directory
2. Install which will distribute the files and run additional scripts:  
   make install
3. Cleanup of temporary files:  
   make clean

If installation complains about missing library, which is already installed, it means that the installed version doesn’t contain API’s to interact with.

API’s are mostly contained in their development kit, so need to install the PACKAGENAME-DEVEL

For example, if gnutls is installed but there are errors about it - install gnutls-devel

### rpm

RPM is redhat package manager.

The RPM package contains:

* The application itself, e.g. app.tar.gz
* List of required dependencies.
* Script files
* Readme
* Changelog
* SPEC file which contains all the setup phases (extract, make etc..)

Since the SPEC is already written for me, it also means that the installation will result in a non-customized installation.

Also, we are dependent on someone to prepare it.

Install command structure: rpm -ivh FILENAME.rpm

i - install

v - verbose

h - show progress bar

Upgrade command structure: rpm -Uvh FILENAME.rpm

U - upgrade/install

v - verbose

h - show progress bar

Remove command structure: rpm -e --nodeps FILENAME.rpm

e - remove

nodeps - ignore removal of dependencies

File structure: app.1.0-el7.x86\_64.rpm

Version

release

Architecture

RPM also manages a database of its installations.

Query whether RPM is installed and in which version: rpm -q PACKAGENAME

Example: amit@clab1 ~> rpm -qa bash

bash-4.2.46-12.el7.x86\_64

Query installed packages: rpm -qa PACKAGENAME

Query for source installation: rpm -qf PACKAGENAME

List all files instead with an RPM: rpm -qf PACKAGENAME

Query the package information: rpm -qi PACKAGENAME

List package files: rpm -ql PACKAGENAME

Verify the package is installed exactly as the package was provided (can also be used to see differences since original setup): rpm -V PACKAGENAME

Query the setup for a specific package which is still not installed: rpm -qip PACKAGENAMEy

Query the scripts executed by the package: rpm -q --scripts PACKAGENAME

Query the changes in the package: rpm -q --changelog PACKAGENAME

Query configuration file for a command: rpm -qc PACKAGENAME

Example:

[amit@clab1 ~]$ rpm -qf /etc/passwd

setup-2.8.71-5.el7.noarch

[amit@clab1 ~]$ rpm -qi setup-2.8.71-5.el7.noarch

Name : setup

Version : 2.8.71

Release : 5.el7

Architecture: noarch

Install Date: Tue 06 Oct 2015 06:21:23 AM UTC

Group : System Environment/Base

Size : 696377

License : Public Domain

Signature : RSA/SHA256, Wed 29 Oct 2014 09:52:39 AM UTC, Key ID 24c6a8a7f4a80eb5

Source RPM : setup-2.8.71-5.el7.src.rpm

Build Date : Wed 29 Oct 2014 09:47:01 AM UTC

Build Host : worker1.bsys.centos.org

Relocations : (not relocatable)

Packager : CentOS BuildSystem <http://bugs.centos.org>

Vendor : CentOS

URL : https://fedorahosted.org/setup/

Summary : A set of system configuration and setup files

Description :

The setup package contains a set of important system configuration and

setup files, such as passwd, group, and profile.

Identify RPM that is used to run a service, example on sshd service:

rpm -qf /usr/lib/systemd/system/sshd.service

Output example: openssh-server-6.6.1p1-12.el7\_1.x86\_64

Red Hat engineers are maintaining different release under the same version. For example, the openssh-server version will remain 6.6.1p1 through the same RHEL version, but the -12 will increase.

### yum

yum (yellowdog updater modified) is using repositories to download and install dependencies alongside the setup files.

The repositories contain XML files and the RPM files for the setup and dependencies.

The information about the web based repositories is maintained under /etc/yum.repos.d/

Repository files end with .repo. example:

[elrepo]

name=ELRepo.org Community Enterprise Linux Repository - el7 ## Repository description

baseurl=http://elrepo.org/linux/elrepo/el7/$basearch/ ## The URL to check against

http://mirrors.coreix.net/elrepo/elrepo/el7/$basearch/

http://jur-linux.org/download/elrepo/elrepo/el7/$basearch/

http://repos.lax-noc.com/elrepo/elrepo/el7/$basearch/

http://mirror.ventraip.net.au/elrepo/elrepo/el7/$basearch/

mirrorlist=http://mirrors.elrepo.org/mirrors-elrepo.el7

enabled=1 # Is the repository enabled

gpgcheck=1 # Whether to check the vendor signature using the gpgkey

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-elrepo.org

protect=0

Verify rpm by importing a KeyFile to my keys list, which will allow to authenticate vendor repository:

rpm --import KeyFile

Repository file can be created manually:

1. Write it
2. Download and install RPM which contains the repository details
3. Using yum-config-manager.  
   Example: yum-config-manager --add=http://REPOSITORY\_URL

Display all available packages: yum list available

Display all installed packages: yum list installed

Display all available and installed packages: yum list all

Display all available group packages: yum grouplist available hidde

Display all installed group packages: yum grouplist installed

Display all available and installed group packages: yum grouplist all

Search: yum search PACKAGE\_NAME

Retrieve info: yum info PACKAGE\_NAME

Install: yum install GROUP\_PACKAGE\_NAME

Upgrade: yum upgrade PACKAGE\_NAME

Downgrade to the previous version: yum downgrade PACKAGE\_NAME

Remove: yum remove PACKAGE\_NAME

Search for the package that will provide a specific command:   
yum what provides COMMAND\_NAME

Update all installed packages (the default command to use in order to update the system): yum update

Install local RPM and use yum to retrieve dependencies:

yum localinstall PACKAGE\_NAME.rpm

Display yum installations history: yum history

Remove using yum repository, for example installation 4: yum history undo 4

Display detailed information about specific history entry, e.g. line 4: yum history info 4

All command can also be done on groups, for example when one wants to install a group of packages. e.g. database: grouplist, groupinfo, groupinstalll etc…

## File systems

link count appears near every file:

-rw-rw-r--. **3** amit amit 18 Dec 6 15:41 file

The link count represents the number of hard links to the actual inode.

inode includes all real information about the file, both its metadata and the actual data blocks.

Files are actually pointers (hard links) to inodes.

If I have more than 1 pointer, than I can delete all pointers except one and still have access to the inode.

Also, pointers can reference each other, since they both use each other to actually represent the inode information.

When deleting a file, what is actually deleted is the pointer to the inode. The inode and the data blocks are not deleted, but their blocks are now available to be overwritten.

Create a hard link to a file: ln FILE\_NAME HARDLINK\_NAME

The file /home/alice/file is not really a file, but a pointer to the inode.

In case bob has permissions on the pointer /home/alice/file but no permissions on /home/alice/ we need to create a hardlink to the inode:

Example: ln /home/alice/file /home/bob/file\_hardlink

This way, bob will have access to the file, without having permissions to /home/alice/

Hard links have limitations, since they are actually pointer to inodes. Limitations:

1. Hard links can only be created in the same filesystem, since it has to work on the same inode table.  
   Since /boot is a different filesystem, the following command will **FAIL**:  
   ln /home/alice/file /boot/file\_hd
2. Hard links can be created only for files, NOT for directories.

Soft links overcome those limitations, by creating pointers to other pointers, and NOT to the inode.

This means that if anything will happen to the destination pointer - the softlink will fail.

create softlink: ln -s HARDLINK POINTER

Softlink is not a shortcut.

When browsing to a softlinked directory - the directory above me will not be the Hardlink directory.

For example:

[amit@clab1 ~]$ ln -s /etc/ conf

[amit@clab1 ~]$ cd conf/

[amit@clab1 conf]$ pwd

/home/amit/conf

[amit@clab1 conf]$ cd ..

[amit@clab1 ~]$ pwd

/home/amit

When using shortcuts - one actually moves to the destination directory.

### Block Device

Block Devices are searchable devices, meaning data access is done randomly.

Sequential access doesn’t require search, for example keyboards.

This means that Block Devices are actually storage devices.

Along the obvious devices (HDD, SSD etc..) - there are other block devices, such as the RAM, RAID arrays, LVM (Logical Volume), Partitions and loop devices (files that file system can be applied on, e.g. iso file).

Up till RHEL6 Ext(2,3,4) was the default filesystem.

Starting from RHEL6 the default filesystem changed to xfs.

**Block** (AKA allocation unit) is the smallest unit available.

**Super Block** - The first block in the disk, includes the file system’s metadata: label, UUID (unique id), total blocks, block size, total inodes, free inodes, free blocks, mount (last time the filesystem was used and where to) etc…  
A super block is created every 8192KB.

**inode** - the metadata of 1 file. It includes: file names / pointers, permissions, owner, modification time, ACL, type, references to the data blocks.

inode directly points to 12 data blocks (when a block equals to 1 KB, then it references 12 KB). In case the file is bigger - the inode references to an **Indirect Block**, which holds additional references to additional data blocks.

The number of references is limited to the size of the filesystem block.

Since in our example a block holds 1 KB, and every reference equals to 4 bytes, the Indirect Block can hold up to 256 references.

In case additional space is required, an additional Indirect Block is created which references to other indirect blocks, each holding 256 references. Since it can also hold up to 256 references, the maximum size it can reference is 2562, which is 64MB.

If this is still not sufficient an additional Indirect Block is referenced by the inode, and that new indirect block holds 256 references to additional 256 indirect blocks which each reference to 256 blocks, so the maximum size is 2563 which is 16GB.

#### Create file system

fdisk is used to create new partitions.

List existing partitions: fdisk -l (in RHEL6 and before: fdisk -l -u -c)

Reload partition table to kernel, if isn’t identified automatically: partx -a DEVICENAME

Example for reloading the partition table for vdb: partx -a /dev/vdb

mkfs is used to create a filesystem, e.g. create ext4 filesystem on /dev/vdb1: mkfs.ext4 /dev/vdb1

blkid displays the superblock information from each device.

Mount filesystem can be done based on the block device label (e.g. media), name (e.g. /dev/vda1) or UUID.

The recommended way to mount to a persistent device is using UUID, then we can be sure that the correct mounts will be maintained across reboots.

Since the same label can be applied on more than 1 block device, while only 1 label to each block device, and the system stops searching for mount points when it finds the first one, we can mount incorrectly.

df -h displays file system information, including used space.

tune2fs is used to configure block devices, e.g. configure a label.

Name /dev/vdb1 FS1: tune2fs -L FS1 /dev/vdb1

Display device info: tune2fs -l /dev/vdb1

Enable ACL on ext2/34 partitions (wasn’t enabled by default on RHEL6 and below): tune2fs -o acl /dev/vdb1

du is used to display disk usage, by default it runs recursively from current location.

Display /mnt/newfs1 size non-recursively: du -hs /mnt/newfs1/

locate is used to find files based on their names.

locate is maintaining an index and this way provides results very fast, but doesn’t provide the latest information written since the last time the indexing was executed.

updatedb & is used to enforce locate the update its indexes with new information.

find is used to locate files, based on their names only, and also has built-in tools to manage files, e.g. compress them.

find is a recursive command which runs ad-hoc and NOT against an index.

Command structure:

find /FOLDERNAME1 /FOLDERNAME2 -maxdepth # SEARCHPARAMETER SEARCHVALUE [[-o] SEARCHPARAMETER SEARCHVALUE]

maxdepth 0 - search only inside the folder’s metadata

maxdepth 1 - search only files in current folder

maxdepth 2 - search 1 level of subfolders

Search parameters can be: name, iname including wildcard, user/owner, type, size (specific size, above +SIZE e.g. above 10MB will be +10M, less than -SIZE e.g. less than 4KB will be -4k), mmin (modified minutes ago), mtime (modified number of days ago), Perm (specific type of permissions, e.g display all files others can execute will be -002).

-o means or

E.g. find all .log files in /home and /tmp which are owned by the currently logged in user $(whoami) or by root:

find /home/ /tmp/ -maxdepth 2 \( -user $(whoami) -o -user root \) -name "\*.log"

If () will not be used then the OR will be:

-user $(whoami) OR -user root -name "\*.log"

which will result in totally different results.

\ is an escape character.

# Shell Scripting

#### Wildcards

|  |  |  |
| --- | --- | --- |
| Character | Meaning | Examples |
| \* | Every character, no matter how long | rm \* will delete all files (other than the hidden ones):  .a  a  a.  a.c  a.log  1.  1.c  1.log |
| ? | Every character that appears only once | rm \*.? will delete:  a.c  1.c |
| [abc\*],[a-z],[0-9a-zA-Z]  Note: MUST write []  [[:upper:]] is the same as [A-Z]  [[:lower:]] is the same as [a-z]  [[:digit:]] is the same as [0-9] | Every character from the list only once, at the target start or end. The list gets its most simplest meaning, for example [abc\*] means a or b or c or \* (not the wildcard, the \* sign specifically).  ‘-’ between characters, means range between character before and after | rm [a-z] will delete:  a.  a.c  a.log |
| [!a-j],[^a-j] | When ! and ^ appear at the beginning, they exclude characters in the list only once, and they also mean that a character must appear there, just not from the list.  If ! or ^ will appear in the middle and not at the beginning or end, they will be treated as search criteria, and not as an exclusion. | rm [!a-z].[!0-9]\* will delete:  1.c  1.log |

In order to delete all files in one command use rm with:

\*.\* all visible files

\*.[!.] all hidden files, and NOT the current folder (“.”) or the one above it (“..”)

#### Output Redirection

E.g. echo A -> filename

No output on screen, will be written to filename

In case we output to file a non-existent output - we will get an error on the screen.

For example, if we run command on non-existent file - get error:

[amit@clab1 ~]$ ll ggg.fff > 1.txt

ls: cannot access ggg.fff: No such file or directory

0 STDIN - Standard Input (Default: Keyboard)

1 STDOUT - Standard Output (Default: Terminal Screen)

2 STDERR - Standard Error (Default: Terminal Screen)

The rest of the outputs do not have a specific number.

Using the numbers we can direct the outputs to different places.

Example on non-existent file - error will now be written to errors.txt:

[amit@clab1 ~]$ ll ggg.fff 1> 1.txt 2> errors.txt

[amit@clab1 ~]$ cat errors.txt

ls: cannot access ggg.fff: No such file or directory

Errors will always be presented before STDOUT, so in case both are written to the same file, STDOUT will overwrite STDERROR.

For example when file exists and NE doesn’t, so the output about file will overwrite the output about NE:

[amit@clab1 ~]$ ll file NE 1>log 2>log

[amit@clab1 ~]$ cat log

-rw-rw-r--. 1 amit amit 0 Dec 2 15:48 file

In order to output all information will use &> to keep both.

&> equals to 1>> and 2>

Example:

[amit@clab1 ~]$ ll file NE 1>>log 2>log

Equals to:

[amit@clab1 ~]$ ll file NE &>log

In order to drop all output command - redirect the output to /dev/null, which is an always empty place.

E.g. output everything to &>/dev/null.

Once set -o noclobber parameter was executed, it will verify files aren’t being overwritten in the current session.

It is used in bash to make sure log files aren’t being overwritten.

In case this parameter was configured, then the command echo A > file will fail and echo A >> file will succeed.

Cancel the option: set +o noclobber

#### Input Redirection

In order to read the file script from the folder above: cat < ../script

ctrl+d - means end of session, will log-off current session

#### Pipes and Filters

##### Pipe

ls | wc -l

The STDOUT of ls is changing to the STDIN of wc -l

If want to pipe errors as well, use |&

Append to file ad-hoc using >| end using ctrl+d to mark done

Example:

[amit@clab1 dir]$ cat >| file3

This is a new line

So writing it

End

[amit@clab1 dir]$ cat file3

This is a new line

So writing it

End

Not all commands can work with | for example rm expects arguments so will not get STDIN.

Therefore the command ls | rm will not work

This can be done using command substitution: rm $(ls)

##### line filters

head [-#]

tail [-#]

grep

grep is short for Get/RegularAxpressions/Print

It displays all the relevant records for the search term

-i - ignore case sensitive

-v - inverse selection, or what is not the search term

-w - display only the specific search term and not when part of

-c - counts number of records that meet the search term

-R - search recursively

-l - list only the file names and not the line that contains the search term

-A # - Show # of lines after search term was found

-B # - Show # of lines before the search term was found, for example, search for an error and display the lines before it to locate the error cause

strings is used to read all strings inside a binary file.

##### Column filters

Columns are separated using delimiters, not always the same ones.

Lines are always separated using the same delimiter - new line.

cut works with columns

Display characters 1-9 in each line: ll | cut -c1-9

Use specific delimiter, for example “:” use cut -d:

Display fields 1 and 3 from /etc/passwd:

cat /etc/passwd | cut -d: -f1,3

tr short for translate is used to replace one character by the other

For example replace : with \t (tab) using tr : “\t”

-d is used to delete characters, for example delete the % character using: tr -d %

awk knows to trim delimiters

-F indicates field delimiter, default is space.

Structure: awk [-Fdelimiter] ‘{print “text” / $#

Since {} have special meaning in bash, we need to add ‘’ to use it as search term.

$# indicates column number to display, e.g. $1 is column 1.

$0 display all line

$(NF) display last column, can also display the one before using $(NF-1)

Using the above commands we can display columns 1 and 3 from /etc/passwd in the following manner:

[amit@clab1 ~]$ head -3 /etc/passwd | cut -d: -f3,1 | tr : "\t"

And can do it more efficiently using awk:

[amit@clab1 ~]$ head -3 /etc/passwd | awk -F: '{print $3"\t"$1}'

awk can be used to display the information in a clearer manner, for example:

[amit@clab1 ~]$ head -3 /etc/passwd | awk -F: '{print "User:"$1"\tUID:"$3}'

User:root UID:0

User:bin UID:1

User:daemon UID:2

Display the disk space used by the partition / from the df -h output:

df -h | grep -w / | awk '{print $(NF-1)}'

Original df -h output:

Filesystem Size Used Avail Use% Mounted on

devtmpfs 487M 0 487M 0% /dev

tmpfs 497M 0 497M 0% /dev/shm

tmpfs 497M 352K 496M 1% /run

tmpfs 497M 0 497M 0% /sys/fs/cgroup

/dev/mapper/centos\_clab1-root 7.6G 1.9G 5.7G 26% /

/dev/vda1 497M 205M 292M 42% /boot

Output of df -h | grep -w / | awk '{print $(NF-1)}' is 26%

uniq removes duplicate lines if they come one after the other. For example, before uniq:

[amit@clab1 ~]$ cat file7

y

y

y

n

n

n

y

y

y

After uniq:

[amit@clab1 ~]$ cat file7 | uniq

y

n

y

sort sorts based on the first column by default. it assumes the column delimiter is space.

-n order by alphanumeric order, used to check numbers as numbers and not text

-r sorts in reverse order

-k# sort based on column # and not the 1st column

-t*delimiter* allows to specify delimiter, for example to use : as delimiter input -t:

Exercises:

Recursively count number of folders under home directory:

ll -R ~ | cut -c1 | grep d -c

Recursively count all files with execute permissions for all users:

ll -R / 2> /dev/null | cut -c4,7,10 | grep xxx -c

**Pipes and Filters Excercise**

1. tail -3 /etc/passwd | awk -F: '{print $1"\t"3}'
2. tail -2 /etc/passwd | tail -1 | awk -F: '{print $1"\t"3}'
3. grep $(whoami) /etc/passwd
4. ls -l ~ | awk '{print $1,$NF}'
5. ll | cut -c9 | grep w -c
6. ls -R ~ | cut -c1 | grep l -c
7. ll | grep "$(date | cut -c5-10)"
8. df -h | grep -w / | awk '{print $(NF-1)}' | tr -d %
9. ll ~ | grep -v total | sort -nk5 | head -3
10. tail /etc/passwd | sort -nk3 -t:
11. ll -d \*[0-9] | cut -c2,5,8 | grep rrr -c

#### Regular Expressions

Bash doesn’t support regex, only wildcards.

There are specific commands, e.g. grep, that support regex.

grep uses a limited set of regular expressions, it is recommended to specify the regex in ‘’.

egrep uses a more extensive library of regular expressions.

|  |  |  |
| --- | --- | --- |
| **Characters** | **Multipliers** | **Line location** |
| . represents any Character  Characters list in [] represents any character from that list or the range it includes [abc.],[0-9],[a-zA-Z0-9]  other than:  ‘-’ represents range  [0-9] equals [0123456789]  ^ represents NOT any of the characters in the list  NOT characters  ^i[a-n]  If a multiplier appears INSIDE a list - it doesn’t represent a multiplier, but the specific character as is | a multiplier defines how many times the character before it should appear  ‘\*’ is multiply the character before it by 0 times through infinity times, no matter how many times. e.g. the search term a\* can be an empty line (a\*0), or apple or aharon but NOT benny  ‘\+’ multiply the character before it by at least 1 time through infinity times  ? multiply the character before it by 0-1  Create specific multiplier, e.g. for ID with 9 digits, not matter which: ID:[0-9]\{9\}  Multiplier in a range, e.g. multiply by any number in the range 3-10 is \{3,10\}  Multiply by at least, for example multiply by 8 and above is \{8,\} | ^ - Line start, e.g. ^B expects to locate files starting with B  $ Line end, e.g. L$ expects to locate files ending with L |

For example the regex ^[a-z][0-9][jkl.]$ represents a 3 character list

**starts** with 1 letter between a-z

**continues** with 1 number, any number

and **ends** with one of the letters j, k or l or the . character

File example:

[amit@clab1 ~]$ cat file

User

User1

User10

SuperUser1A

Display only lines starting with User[0-9] in file:

[amit@clab1 ~]$ grep ^User[0-9] file

User1

User10

Display only lines starting with User[0-9] and ending with 1 number only in file:

[amit@clab1 ~]$ grep ^User[0-9]$ file

User1

|  |  |  |
| --- | --- | --- |
| Example line: | ^.\*$ | ^a\*$ |
| me | will find line | Will not find the line since it doesn’t start with a |
| apple | will find line | Will not find the since it doesn’t end with a |
| a | will find line | will find line |
| [empty line] | will find line | will find line as if it was a multiply of a\*0 |

list all lines starting with d, meaning list all directories:

ls -l | grep ‘^d’

Display lines starting with a letter and ending with number in datafile:

grep ‘^[A-za-z]\*[0-9]$’ datafile

Display lines starting with ‘.’ and has at least some second character which is not ‘.’, which is actually a representation of all files except the current dir ‘.’ and the one above it ‘..’ when running ls -l: ls -a | grep '^\.[^.]'

() represents a string which is multiplied as is, example:

[amit@clab1 dir]$ cat file

vi

viable

viableable

viabl

[amit@clab1 dir]$ grep "^vi\(able\)\*$" file

vi

viable

viableable

If there are no (), then it will multiply the last character only, in this case ‘e’:

[amit@clab1 dir]$ grep "^viable\*$" file

viable

viabl

**Exercise 8, page 166**

1. grep "^d" datafile
2. grep "^d.\*[0-9]$" datafile
3. grep "^[-l.]" datafile
4. grep ou datafile
5. grep "[0-9]" datafile
6. grep "^-.\*n" datafile
7. grep "^.\{1,10\}$" datafile
8. grep "^[^.]\{7\}$" datafile
9. grep . datafile
10. grep "^[^ \t]" datafile
11. grep -l "^t.\*[0-9]$" \*[0-9]
12. grep ^$ datafile -c
13. ll -R | grep "^[d-][^-]\{9\}"

#### Advanced find

Command structure:

find /FOLDERNAME1 /FOLDERNAME2 -maxdepth # SEARCHPARAMETER SEARCHVALUE [[-o] SEARCHPARAMETER SEARCHVALUE] [-exec COMMAND\_TO\_EXECUTE\_ON\_RESULTS {} \i]

maxdepth 0 - search only inside the folder’s metadata

maxdepth 1 - search only files in current folder

maxdepth 2 - search 1 level of subfolders

Search parameters can be: name, iname including wildcard, user/owner, type, size (specific size, above +SIZE e.g. above 10MB will be +10M, less than -SIZE e.g. less than 4KB will be -4k), mmin (modified minutes ago), mtime (modified number of days ago), Perm (specific type of permissions, e.g display all files others can execute will be -002).

-o means or

-exec - execute commands

-ok - asks to confirm executing the command

example - find all files ending with .log which are owned by the current user or root and delete them:

find /home /tmp -maxdepth 5 \( -user $(whoami) -o -user root \) -name "\*.log" -exec rm -f {} \;

The above command creates the following command and executes it:

rm -f /home/student/a.log; rm -f /home/student/b.log; rm -f /tmp/root.log

Since BASH runs before find and it doesn’t support regex - results are different when executing find with \*. When using "", it tells BASH not to interpret and leave it as is for find. In order to use regex - use the -regex option.

[amit@clab1 ~]$ find -name \*.log

./dir/a.log

./a.log

[amit@clab1 ~]$ find -name "\*.log"

./dir/a.log

./dir/b.log

./dir/c.log

./dir/d.log

./a.log

**Find exercise**

1. find /tmp/ -user $(whoami) -type d 2>/dev/null | wc -l
2. find ~ -type f 2>/dev/null | wc -l
3. find ~ -mmin -480 2>/dev/null | wc -l
4. find ~ -size +1M -exec du -h {} \; 2>/dev/null
5. find ~ -name "\*.c" 2>/dev/null
6. find ~ -name "\*.temp" -exec rm {} \; 2>/dev/null
7. dd if=/dev/zero of=~/large.log bs=1M count=32
8. find ~ -size +20M -name "\*.log" -exec gzip {} \; > /dev/null
9. find ~ -type d -user $(whoami) -exec grep total -R {} \; 2>/dev/null
10. du -m $(find ~ -type d -user $(whoami) -exec find {} -maxdepth 1 -size +2M \; 2>/dev/null)

dd is a data duplicator command on the block level, that copies from the if (input file) to the of (output file).

bs sets block size.

count is used to specify how many times to repeat the action and is optional. Will repeat 1 time only if not specified.

It is used for data duplications.

/dev/zero is a file with infinite 0’s, used for thick provisioning or delete data before sending to recycle.

For example, restore valid MBR to the first 512 bytes of a faulty disk using:

dd if=mbr.img of=/dev/sda bs=512 count=1

dd can also be used to check disk write speed.

For security purposes, it is advised to use /dev/random and /dev/urandom based duplications, and not dd which has a known pattern. /dev/random depends on user actions, which aren’t predictable but there aren’t many, so one can use /dev/urandom which contains much more random information.

#### Variables

Variables are memory location, identified by name and include some content.

For example, set the variable dog with the content snoopy:

[amit@clab1 ~]$ dog=snoopy

[amit@clab1 ~]$ echo $dog

snoopy

In order to read a variable use $VARIABLENAME

Export variables to child processes, for examples when executing a script file using BASH: export VARIABLENAME

Can create and export: export var2=value2

Unset variable: unset VARIABLENAME

Example:

[amit@clab1 ~]$ echo $dog

snoopy

[amit@clab1 ~]$ unset dog

[amit@clab1 ~]$ echo $dog

Display all variables, exported and those created ad-hoc: set

Display all exported variables: env

**Important variables**

PATH includes all folders which include commands that can be executed from any location. BASH interpretation order - checks the command based on the following order:

1. alias to some other command
2. local file in the current location
3. dictionary word (reserved word for BASH, e.g. echo or cd or exit)
4. included in some folder included in PATH

Add folder to PATH, e.g. add /home/student/old by executing: PATH=/home/student/old:$PATH

HOME includes the user’s home directory.

UID displays current user UID

EUID displays the effective UID, meaning the UID of the person which started the session and not su’d to. E.g. session started with user amit with UID 1000, then su’d to root and executed echo $EUID - the result will be amit’s UID and not root’s.

SHELL displays default user shell

0 displays the the current shell the user is logged in to

$ returns the PID of the current running process.

? returns last exit status, Windows error level equivalent

PS1 - prompt string

PS2 - the result of the second Prompt String, used when the command wasn’t written properly and a new prompt opened

PWD current location

OLD\_PWD previous location

HISTFILE file to which commands history is written

HISTSIZE displays the size of the history file size

history displays command history. Usage:

Interactive: history, arrows up and down, ctrl+r search for the last command

Non-Interactive:

!! last command executed

!# the command # as appear in the history file

!-# command executed before -# times

!str last command started with the specified string. For example:

[amit@clab1 ~]$ cat devnull

[amit@clab1 ~]$ !c

cat devnull

!$ last argument, example:

[amit@clab1 ~]$ cat devnull

[amit@clab1 ~]$ more !$

more devnull

**Quoting**

"" makes the text between "" to lose its meaning, other than:

"

$

!

\ removes the meaning of the characters which come after it and adds special meaning for control characters, e.g. \n \t

For example, use \ to make $ lose its meaning:

[amit@clab1 ~]$ echo " you owe me $5"

you owe me

[amit@clab1 ~]$ echo " you owe me \$5"

you owe me $5

'' is for strong quoting, it loses the meaning of everything, other than ' which ends the quote.

**Alias**

Alias is used to create aliases to commands.

For example, set c as an alias to clear: alias c=clear

Set rm as rm -i by default: alias rm='rm -i'

In order to ignore alias, use \ before the alias. e.g. \c or \rm

Cancel alias: alias alias=

Example: unalias c

**Profile files (Environment files)**

Files that determine the user’s profile.

|  |  |  |
| --- | --- | --- |
|  | **Global** | **Personal** |
| **Login** | /etc/profile is the login script for all users, uses the folder /etc/profile.d which includes additional configuration files | ~/.bash\_profile is a hidden file inside the user’s home directory |
| **Shell** | /etc/bashrc is the default bash startup script | ~/.bashrc |
| **Logout** |  | ~/.logout |

Execution order:

1. /etc/profile
2. ~/.bash\_profile
3. ~/.bashrc which calls to and after completing returns to ~/.bash\_profile since it was called by it.
4. /etc/bashrc which executes and since it was called by ~/.bashrc it returns to it.

so the actual order is:

1. /etc/profile
2. ~/.bash\_profile
3. ~/.bashrc
4. /etc/bashrc
5. ~/.bashrc
6. ~/.bash\_profile

**Advanced BASH exercise**:

1. mkdir ‘Clint Eastwood’
   1. touch Clint \Eastwood\Million$Baby
2. PS1='[$(whoami)@$(pwd)]$'
3. Solutions:
   1. alias run="chmod a+x"
   2. alias files="ls -lR | grep ^-"
   3. alias links="ls -lR | grep ^l"
   4. alias large="find -size +2M -exec du -h {} \; 2>\dev\null"
   5. alias rm="mv -t ~/.recycle/"
   6. alias disk='df | grep /$ | awk "{print \$(NF-1)}" | tr -d %'
4. alias >> ~/.bashrc
5. Done
6. echo $HISTFILE
7. cat !$
8. HISTSIZE=2000
9. echo HISTSIZE=2000 >> ~/.bash\_profile

#### Scripts

Mostly starts with indication about the interpreter:

#!/bin/bash

and a general comment about the script

# This script is intended to….

**Scripts execution**

|  |  |  |
| --- | --- | --- |
| **Execution command** | **Script path** | **Required permissions** |
| /path/to/script.sh  In case in the script directory:  ./script.sh  The reason for adding ./ is to defend against unintentional execution. | The script session is running inside a child BASH session.  Therefore, variables created by the scripts are not maintained when completing the script execution and returning to the parent BASH session. | rx  Note: Files written using vi are created by default without execution permissions and those are needed to be added manually. |
| . /path/to/script.sh  In case in the script directory:  . script.sh  the . at the beginning indicates a binary execution file. | . script.sh  The script runs inside the same BASH session, so variables created by the script, are maintained in the execution session. Used for example to execute the BASH profile scripts. | r |
| bash /path/to/script.sh  In case in the script directory:  bash script.sh  useful flags:  -v - verbose, display script text while execution  -x - writes to console in debug log level | The script session is running inside a child BASH session.  Therefore, variables created by the scripts are not maintained when completing the script execution and returning to the parent BASH session. | r |

**Arguments / Parameters**

Information passed to the scripts.

It is displayed using $.

$number returns the argument with that number.

$0 variable represents the script name.

BASH supports single number arguments, so $10 will be parsed as $1. In order to return the 10th argument - execute ${10}

$# returns the number of arguments

$\* returns all arguments

$@ returns all arguments as 1 argument

Example script:

#!/bin/bash

#Sample script

echo $2 $1

Example execution:

[amit@clab1 ~]$ . script.sh Amit Arik Benny

Arik Amit

First the script echos $2 (Arik) and then $1 (Amit). Benny turns to $3, but it isn’t actually used by the script.

If the script will change to:

echo $#

The number of arguments will be returned:

[amit@clab1 ~]$ . script.sh Amit Arik Benny

3

If the script will change to:

echo $\*

All arguments will be returned:

[amit@clab1 ~]$ . script.sh Amit Arik Benny

Amit Arik Benny

If the script will change to:

echo $@

All arguments will be returned, but unlike $\* they will be converted to 1 argument (running with $\* returns A B C while $@ actually returns “A B C”:

[amit@clab1 ~]$ . script.sh Amit Arik Benny

Amit Arik Benny

**Reading Variables**

In order to be able to interact with the person executing the script and act according to his answers, use read -p “text to be prompted” var1 var2

Example script:

#!/bin/bash

#Sample script

read -p "enter number of seconds and program to execute: " sec prog

sleep $sec

$prog

Example execution:

[amit@clab1 ~]$ bash script.sh

enter number of seconds and program to execute: 5 ls

a.log file file7 large.log large.log.gz rt. script.sh

!# as is returns the last argument

**Parameters exercise**

1. echo "We sent $# args into $0. total args are $\*. last arg is: ${!#}"
2. size=$(ls -l "$1" | awk '{print $5}')  
   echo "The size of $1 is $size bytes
3. owner=$(ls -ld "$1" | awk '{print $3}')  
   chown -R $owner "$1"
4. echo -n "Please state the name of the file from $1 you wish to apply execution permissions on: "  
   read filename  
   chmod a+x "$1"/"$filename"

**Conditions**

if has a couple of structures:

1 condition:

if CONDITION

then

COMMAND

fi

1 condition and what to do if condition isn’t met:

if CONDITION

then

COMMAND

else

OTHER COMMANDS

fi

1 condition and another condition if the 1st condition isn’t met etc:

if CONDITION

then

COMMAND

elif CONDITION2

then

OTHER COMMAND

else

OTHER COMMAND

fi

The CONDITION is called test command, because it also has a syntax with the word test.

Structure: Value2Check ComparisonParameter(= / != etc…) Value2CheckAgainst

Full information can be found at the man pages of test

File tests:

-e exists

-f file

-d directory

-L softlink

-r readable for the user executing the command

-w writable for the user executing the command

-x executable for the user executing the command

-nt newer than

-ot older than

-ef equivalent file

String tests:

-z zero length

-n not zero length, can also be written as !-z

String1 = String2 verify the strings are the same

> greater than, lexical comparison, this is why 7 < 10 is false since it compares character by character so it first compares 7 to 1

>= greater than or equal

!= not equal

~= similar, meaning the wildcard used in string1 matches string2

Number tests:

-eq equal

-gt greater than

-ge greater than or equal

-lt less than, this is 7 -lt 10 here is true, unlike with the strings comparison (7 > 10)

-le less than or equal

-ne not equal

Example string:

#!/bin/bash

#Treats the file (given by the argument) according to its type

if [ -n "$1"] # tests the given file isn’t zero length

then

if [ -e "$1" ] # tests the given file exists

then

if [ -f "$1" ] # tests the given file is a file

then

cat "$1"

elif [ -d "$1" ] # tests the given file is a directory

then

ls -l "$1"

else

echo "$1 isn't a file nor a directory"

fi

else

echo "$1 doesn't exists"

fi

else

echo "No argument was sent"

fi

**Conditional Execution**

Exit status indicates the status of the executed command

0 is success

1 etc.. is some sort of failure

The exit status can be read using $?

Success status example:

[amit@clab1 ~]$ ping 8.8.8.8

PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.

64 bytes from 8.8.8.8: icmp\_seq=1 ttl=52 time=71.0 ms

1 packets transmitted, 1 received, 0% packet loss, time 0ms

rtt min/avg/max/mdev = 71.095/71.095/71.095/0.000 ms

[amit@clab1 ~]$ echo $?

0

Failure status example:

[amit@clab1 ~]$ cat gggggjklll

cat: gggggjklll: No such file or directory

[amit@clab1 ~]$ echo $?

1

Command structures of Conditional Execution:

COMMAND1 && COMMAND2

&& indicates execution only if COMMAND1 succeeded to run (returned exit code equal to 0)

COMMAND1 || COMMAND2

|| indicates execution only if COMMAND1 failed to run (returned exit code bigger than 0)

For example, we have an alias to check the disk space usage of /boot:

alias disk='df | grep /boot | awk "{print \$(NF-1)}" | tr -d %'

Then we can check the disk status and if it is greater than 80% send an email about it:

[ $(disk) -gt 4 ] && df -h | mail -s " Disk /boot almost full" [test@test.com](mailto:test@test.com)

**Conditions exercise**

1.

#!/bin/bash

#Verifies specific file and applies execute permissions on it

if [ -d "$1" ]

then

echo -n "Please enter a file name: "

read file

if [ -e "$1"/"$file" ]

then

chmod a+x "$1"/"$file"

ls -l --color=auto "$1"/"$file"

else

echo "Cannot find $1/$file"

fi

else

echo "invalid directory specified"

fi

2.

#!/bin/bash

#This script is intended to locate process name provided and check how many times it runs

num=$(ps -e | grep -c "$1")

if [ "$num" -eq 0 ]

then

echo "$1 is not running"

elif [ "$num" -le 10 ]

then

echo "$1 is running $num times"

else

read -p "$1 is running $num times, would you like to kill it?[y/n] " answer

if [ "$answer" = "y" ] || [ "$answer" = "Y" ] # || indicates OR

then

killall "$1"

else

echo "Over 10 instances of $1 are running, please check"

fi

fi

**Loops**

**While loop structure**:

while CONDITION

do

COMMANDS

done

Example 1:

#!/bin/bash

#This script prints all given argument using the while loop

while [ $# -gt 0 ]

do

echo $1

shift #shifts all arguments one place left so the last argument called is deleted from the queue

done

Example 2:

#!/bin/bash

#The script prints only non-empty lines

cat "$1" | while read my\_line # while automatically reads input line by line, my\_line indicates the current line read

do

[ -n "$my\_line" ] && echo "$my\_line"

done

Can also be accomplished:

u

while read my\_line

do

[ -n "$my\_line" ] && echo "$my\_line"

done < "$1"

**CANNOT** be done using the following example, since it will result in infinite loop, as it always returns to the while command:

while read my\_line < “$1”

do

[ -n "$my\_line" ] && echo "$my\_line"

done

**for loop structure**:

for VARIABLE [ in VALUELIST ]

do

commands

done

Example 1:

#!/bin/bash

#The script does for loop

for fruit in Apple Mango Orange

do

echo " I like ${fruit}s"

done

Example 2:

#!/bin/bash

#The script shows info only on runnable files

for file in "$1"/\*

do

[ -f "$file" ] && [ -x "$file" ] && ls -l "$file"

done

Example 3 - displays how [ in VALUELIST ] is optional:

#!/bin/bash

#The script shows all inserted arguments

for my\_arg

do

echo $my\_arg

done

What for does when no VALUELIST is specified is to use $@ as VALUELIST:

[amit@clab1 ~]$ bash -x for\_loop3.sh A B C D

+ for my\_arg in '"$@"'

+ echo A

A

+ for my\_arg in '"$@"'

+ echo B

B

+ for my\_arg in '"$@"'

+ echo C

C

+ for my\_arg in '"$@"'

+ echo D

D

**Loops exercise**

1.

#!/bin/bash

#Reports absent users to log file

cat "$1" | while read user

do

if grep "^$user:" /etc/passwd

then

cp ~/file ~$user

else

echo "$user wasn't found on $(date)" >> absent.log

fi

done

2.

#!/bin/bash

#checks whether a gz file exists and adds extension if missing

for file in "$1"/\*

do

if [ -f "$file" ] && echo $file | grep -v "\.gz" &>/dev/null

then

file $file | grep "gzip" &>/dev/null && mv -T "$file" "${file}.gz"

fi

done

**sed**

sed is stream editor

Allows to execute commands on input

Structure: sed [-nei] COMMAND INPUT

sed command will execute on all input lines.

Commands:

p print

d delete

a STRING append the STRING under the relevant line

i STRING insert the STRING above the relevant line

w file write the relevant lines to file

s/OLDTEXT/NEWTEXT/gi replace OLDTEXT with NEWTEXT in all relevant lines. /gi g stands for global (all entries, not only at line start) and i stands for ignore case

Examples: replace the first - with f: ll | sed s/-/f/

To use sed only on specific lines execute: sed [-nei] LINE\_IDENTIFIER sed INPUT

For example to delete the 1st line: ls | sed 1d

Delete the lines starting with the word ‘total’: ls | sed /^total/d

Identify all lines starting with - and replace - with f: ll | sed /^-/s/-/f/

Execute more than 1 expression using -e:

ll | sed -e '/^-/s/-/f/' -e '/^total/d'

Or concatenate the lookup strings using ;:

ll | sed -e '/^-/s/-/f/;/^total/d'

-n silent mode, print only what was requested

For example, print only the lines starting with d: ll | sed -n /^d/p

-i insert changes

For example, replace bash with csh in users: sed -i s/bash/csh/ users

Execute sed between lines:

sed [-nei] LINE\_IDENTIFIER1 LINE\_IDENTIFIER2 sed INPUT

For example, display all lines between line with ERROR and line with END:

sed -n /ERROR/,/END/p file

Display line before ERROR and after END:

sed -n /"$(grep ERROR -B1 file | head -1)"/,/"$(grep END -A1 file | tail -1)"/p file

# RH134

## Kickstart

**Anaconda** manages the operating system installation. Possible resources:

1. **Media**
   1. **Manual**
   2. **Automated** using a **kickstart** file (equivalent to Windows Answer file) - need to let the Media know the path to the kickstart file
2. **Network** - client boots from PXE and using a DHCP server gets the TFTP address of the install server (DHCP “next server” option) and the filename that should be retrieved.
   1. **Manual** - only the Media is loaded automatically
   2. **Fully automated** using a kickstart file

The install server contains /var/lib/tftpboot directory containing:

pxelinux.0 - a small application which uses configuration taken from:

pxelinux.cfg directory, which contains either a default configuration file or custom configuration file for specific client using its IP or MAC address or GUID.

The configuration file contains references to *vmlinuz* (linux kernel, here used temporarily for the installation process, it is not the kernel which will be installed), *initrd.img* (contains modules, e.g. drivers, required by some systems) and *inst.stage2* which contains the path containing to the relevant OS installation media.

The **kickstart** file contains the required configuration that will be set during the installation, e.g. root password hash, partitions, packages (@ indicates group of packages, - indicates to avoid installing specific packages, e.g. when installing the Internet Browsers do not install Firefox), pre-install scripts, post-install scripts etc…

A kickstart file can be created:

* **Manually**
* Using the file **/root/anaconda-ks.cfg** created automatically during the installation
* **Kickstart Conifgurator** software

## Scheduling

at is used for one time scheduling.

Structure: at [-q a/b/c] When

Example to schedule in 4 hours: at now+"4 hours"

> COMMAND

List using atq or at -l

-q is used to create queues for priorities.

-c is used to display the actual script that will run. For example to display the script for job 4: at -c 4

at -r # or atrm # is used to delete tasks based on their job id.

batch is a subcommand of at running tasks only when the load on the system is low.

wall is used to write something to all open consoles.

cron is used to schedule ongoing repetitive tasks.

There is a crontab for the system: /etc/crontab and for each user /var/spool/cron/USERNAME

Add users to /etc/cron.deny to disable users from creating cron jobs.

If user appear at both /etc/cron.deny and /etc/cron.allow - he will be allowed to schedule jobs. It is used to block everyone, by adding \* to /etc/cron.deny and specifying the usernames which are allowed at /etc/cron.allow

crontab is used to create cron jobs.

-e - edit

-l - list

-r - remove

crontab structure:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Minute** | **hour** | **day** | **month** | **weekday** | **Command** |
| **Possible entries** | 0-59 | 0-23 | 1-31 | 1-12  Or:  Jan-Dec | 0-7  0=7=Sunday  Or:  Sun-Sat | CMD |

Minute can be represented as a number and also as calc, for example every 5 minutes is \*/5 (every number divided by 5 without a remainder)

Example: 1 2 3 4 5 CMD

Will run on 02:01 AM on April 3rd if the weekday is Friday.

Example: \*/5 2-4,14-16 5,6 7-10 \* CMD

Will run every 5 minutes between 14:00-16:59 on the 5th and 6th of months Jul-Oct no matter which weekday.

View system cron scripts under the /etc/cron… directories: daily, hourly, monthly, weekly.

Can add scripts so they will also run during those events

anacron is used to execute tasks upon incident.

Configured using /etc/anacrontab

|  |  |  |  |
| --- | --- | --- | --- |
| Period in days | delay in minutes | job-identifier | command |
| 1 - daily  7 - weekly  @monthly - monthly | How much time to wait after system completed boot (crond started), used to avoid running multiple tasks in the same time | job id | CMD |

Main configurations:

RANDOM\_DELAY- maximal random delay added to the base delay of the jobs

START\_HOURS\_RANGE - hours range in which running tasks is allowed

anacron checks the file /var/spool/anacron/cron… to see when was the last time it ran and decide whether to execute or not.

For example, the last time cron.daily were executed was on Jan 15, 2016:

[root@clab1 ~]# cat /var/spool/anacron/cron.daily

20160115

If one wants to disable running anacron tasks - write the date manually to /var/spool/anacron/cron…

List files with scheduled tasks per installed package: rpm -qc PACKAGENAME

systemd-tmpfiles handles the rest of the repetitive tasks: managing temp directories.

The command systemd-tmpfiles allows to mark files/folders as temp directories.

The full configuration can be found under tmpfiles.d

Allows to create folders if they do not exist, create block devices etc…

Example for configuration (from man tmpfiles.d):

#Type Path Mode UID GID Age Argument

d /run/user 0755 root root 10d -

L /tmp/foobar - - - - /dev/null

Age - not touched for the time mentioned, e.g. 10d means not touched for the last 10 days, will delete the folder and its content and recreate it.

3 default scripts directories:

/usr/lib/tmpfiles.d/

/run/tmpfiles.d/

/etc/tmpfiles.d/

/run is a file system running on the RAM. Therefore, programs write its tmpfiles configuration to /run/tmpfiles.d/

The order in which tasks are loaded:

1. /usr/lib/tmpfiles.d/
2. /run/tmpfiles.d/ - if there is a conflict will override the files from /usr/lib/tmpfiles.d/
3. /etc/tmpfiles.d/ - if there is a conflict will override the files from /usr/lib/tmpfiles.d/ and /run/tmpfiles.d/  
   It is recommended ONLY to edit files in this directory and not change the scripts in the other directories.

## Access Control List

ACL allows to extend the permissions set to more than the regular ugo and rwx options.

For example, allow a specific user rwx, in addition to the owning user.

Display ACL: getfacl FILENAME

Edit ACL: setfacl -m FILENAME

-m - modify

For example, add rwx permissions for the root user on nohup.out:

# Original ACL

root@clab1 ~]# getfacl nohup.out

# file: nohup.out

# owner: root

# group: root

user::rw-

group::---

other::---

# Modify the ACL to include rwx for root

[root@clab1 ~]# setfacl -m u:root:rwx nohup.out

# Modified ACL, will remove the group entry and insert the file’s mask

[root@clab1 ~]# getfacl nohup.out

# file: nohup.out

# owner: root

# group: root

user::rw-

user:root:rwx

group::---

mask::rwx

other::---

Once an ACL entry exists the 11 character in the ls -l will be ‘+’:

# Before

[root@clab1 ~]# ll nohup.out

**-rw-------.** 1 root root 11419835 Dec 9 16:14 nohup.out

# After, note the group section was replaced with the mask info

[root@clab1 ~]# ll nohup.out

**-rw-rwx---+** 1 root root 11419835 Dec 9 16:14 nohup.out

**mask** indicates the maximum permissions that can be set on a file.

Allows to control the maximum permissions files can get.

The default is rwx.

Doesn’t impact the user and other, only the group and the users with ACL entry.

This means that if the mask is read-execute only, the effective permissions the group and the users with an ACL entry can get is r+x, for example:

# Set mask read-execute

[root@clab1 ~]# setfacl -m mask:rx nohup.out

# Display effective permissions

[root@clab1 ~]# getfacl nohup.out

# file: nohup.out

# owner: root

# group: root

user::rw-

user:root:rwx #effective:r-x

group::---

mask::r-x

other::---

Every change using setfacl resets the mask to the same acl, other than when setting the mask specifically using it.

ACL can be used to set inheritance on sub-directories.

For example, to set the **default** settings for the user nobody to none on all files under the dir1 directory execute: setfacl -m **d**:u:nobody:- dir1/

When adding default permissions, recursive is implied, and the new default set is applied recursively.

Conditional execution is indicated using capital X.

It means that in case files are already configured with execute permissions, those permissions will remain, but no new execute permissions will be added.

The newgrp command is used to change the current group ID during a login session.

For example, if my default group is amit and I want to change it for the current session to teachers, I need to execute: newgrp teachers

The file system ext4 doesn’t support ACL by default and requires to be mounted with the acl option.

Remove acl entry using setfacl -x without mentioning the specific permissions, e.g. remove the acl entry of alice from docs: setfacl -x user:alice docs

Import acl entries from a file:

* Export acl: getfacl docs > acl\_backup
* Delete acl before import: setfacl -b docs2
* Import acl: setfacl -M acl\_backup docs2

## SELinux

Security Enhanced Linux.

Manages permissions based on the context in which commands are executed.

For example, the user apache, while running httpd, is allowed access to /var/www/html but disallowed to access /tmp

This way, malicious hackers that might take control over the httpd , will not be able to take control over the whole system.

SELinux checks the context configured on the target. For example httpd\_sys\_content\_t is the type set on /var/www/html/index.html . It allows access between similar types, so Apache running as httpd\_t can read /var/www/html/index.html

**Types**:

* **Targeted** - most common, enforced on targeted processes.
* **Minimal** - same as targeted but on specific processes
* **MLS** - Multi Layer/Level Security, checks not only the user and what he wants to do, but also what he wants to do, what role does the user have and his permissions. it is rarely used.

**Security levels**:

* **Enforcing** - enforces the SELinux policy
* **Permissive** - does not enforce, but writes to Logs
* **Disabled** - SELinux not working at all

getenforce - display enforcement status

setenforce - change enforcement mode for the current session, to change mode permanently edit /etc/sysconfig/selinux

Disable SELinux on the fly for testing (doesn’t require testing): setenforce 0

Enable back SELinux: setenforce 1

To totally disable edit /etc/sysconfig/selinux and restart the server

semanage - used to manage the SELinux policy

ls -Z - display SELinux context for file/folder

ps -eZ - display context for running processes

[amit@clab1 ~]$ ls -Z file

-rw-rw-rw-. amit amit unconfined\_u:object\_r:user\_home\_t:s0 file

Context:

user:role:type:security\_level

The default type is **Targeted**.

The only characteristic used in **Targeted** mode is type (others are used in **MLS** only).

semanage fcontext -l - manage the file contexts

To display specific context, use grep, for example on /var/www/html:

semanage fcontext -l | grep “/var/www/html”

Copy the context from one file to another using chcon:

chcon --reference=/home/amit/file /home/amit/file7

Change the SELinux content type, for example change the file context on /custom recursively to httpd\_sys\_content\_t:

semanage fcontext -a -t httpd\_sys\_content\_t ‘/custom(/.\*)?’

Remove SELinux content type added manually using -t:

semanage fcontext -d -t httpd\_sys\_content\_t ‘/custom(/.\*)?’

In order to apply the context changes done on the parent folder using chcon or semanage:

restorecon -Rv DIRECTORYNAME

SELinux messages are written to /var/log/audit/audit.log

setroubleshoot is used to read the SELinux manages in a human-readable format.

sealert is used to read SELinux message in details: selinux -l MESSAGE\_ID

setsebool is used to allow well known contexts.

Display all available context with booleans: semanage boolean -l

Change boolean for specific context (persists across reboots): setsebool -P CONTEXT 0/1

For example, display httpd\_read\_user\_content boolean setting:

[root@localhost ~]# semanage boolean -l | grep httpd.\*user

httpd\_read\_user\_content (off , off) Allow httpd to read user content

Change boolean temporarily:

[root@localhost ~]# setsebool httpd\_read\_user\_content 1

Display httpd\_read\_user\_content boolean setting:

[root@localhost ~]# semanage boolean -l | grep httpd.\*user

httpd\_read\_user\_content (on , off) Allow httpd to read user content

Change boolean permanently:

[root@localhost ~]# setsebool -P httpd\_read\_user\_content 1

Display httpd\_read\_user\_content boolean setting:

[root@localhost ~]# semanage boolean -l | grep httpd.\*user

httpd\_read\_user\_content (on , on) Allow httpd to read user content

When adding services that do not have a mandatory SELinux policy preconfigured by red hat, need to configure them first as unconfined so that SELinux will allow it to run without restrictions, and after learning what it does create a custom policy for it.

Can also setenforce 0 for a limited time and see what is required by the new service using the audit.log entries.

To display all semanage context policy (need to install setroubleshoot using yum first): semanage fcontext -l

For the test we need to know the SELinux for:

**Apache**:

http\_sys\_content\_t (html files)

http\_sys\_script\_exec\_t (cgi scripts)

**Samba**:

samba\_share\_t (shared folders)

samba\_enable\_home\_dirs

**Directory used by a couple of services**:

Public\_content\_t (allows to share a directory between a couple of services)

## LDAP & IPA

In order to work with **LDAP** use the following packages:

1. SSSD - LDAP client, configure either using CLI or TUI
2. authconfig-gtk - GUI for SSSD
3. krb5-workstation - to support password encryption using kerberos

The system-config-PACKAGENAME opens a TUI for supported packages.

Verify connectivity to domain: getent passwd USERNAME

**IPA** stands for *Identity Policy Authorization* (equivalent to the Windows *Group Policy*).

It allows to enforce permissions set by the domain controller on the authenticated entity, for example allow to restart services or access a directory.

Configure IPA Client to work with a DC, after downloading using yum: ipa-client-install

The Windows Group Policy will enforce basic settings.

By extending the GP schema can control more Linux-specific settings.

## Block Devices

Under **/dev** we have **Character** and **Block** devices.

Blocked devices types: Disk, RAM, loop, LV, RAID

Block devices are indicated by a starting b in their ls output.

Different disks differentiate by a different letter: sda, sdb, sdc....

Different partitions differentiate by a different number: sda1, sda2, sda3....

Each disk has a Master Boot Record, mostly sized at 512 bytes.

The **MBR** consists of:

* **Bootloader** sized at 446Bytes
* **Magic Number** (checksum) sized 2B
* **Partition Table** sized 64B, which includes the partitions info (start and end cylinder, primary etc…). Each partition info is sized 16B, so a maximum of 4 Primary partitions can be created.   
  A **Primary partition** is ready to be configured with a filesystem.  
  If we want more than 4 partitions - we need to use an **extended partition**.  
  An extended partition cannot be configured with a regular file system, as it only contains metadata about the **logical partitions**.  
  The logical partitions can be installed with a file system.  
  When creating an extended partition, allow it to consume all free space left on the disk to fully utilize it.

BIOS can only work with a Primary partition to boot an operating system from.

fdisk is used to edit the disk partition table for MBR disks.

gdisk is used to edit the disk partition table for GPT disks. GPT stands for GUID Partition Table. GPT is located at the end of the disk, normally sized at 1MB, which allows it to support more primary partitions and partitions larger than 2TB.

partprobe is used to update the kernel with the changes configured by fdisk.

parted is a more updated command, that does the same as fdisk/gdisk and supports partition names.

partx -a /dev/DISKNAME is used if partprobe isn’t working.

mkfs is used to create file system on the new partition.

blkid is used to display the unique block ID.

tune2fs -l displays partition metadata (magic number, UUID etc…)

Each file system includes a **superblock**, which contains information about:

* File system type
* Size
* Status
* Information about other metadata structures

Mount partition:

1. Make the directory to mount to, e.g: mkdir /mnt/fs1
2. Mount using mount: mount PARTITIONNAME DIRECTORY2MOUNT  
   Or: mount UUID=”UUID\_String” DIRECTORY2MOUNT  
   Example: mount /dev/vdb1 /mnt/fs1/

To mount partitions permanently edit /etc/fstab:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Device/UUID/Label** (reads from the superblock and this way not losing or mounting wrongly) | **Mount Point** | **File system type** | **Mount Options** (e.g. load automatically or not) | **Backup including dump** | **fsck** |
| UUID=b1acf8a9-3310-46ba-ac65-9bee0cbe07a4 | /boot | xfs | defaults | 0 | 0 |
| /dev/mapper/centos\_clab1-root | / | xfs | defaults | 1 | 1 |
| LABEL=Media | /mnt/fs1 | ext4 | noauto | 0 | 0 |

mount -a is used to mount partitions based on the /etc/fstab

swapon -a is used to mount swap space based on the /etc/fstab

Swap space is used by the operating system to move less used pages from the RAM.

This space resides on the disk, so querying it takes longer than querying pages in the RAM.

Swap space can be done using a file on the disk or using a separate partition dedicated for swapping.

Best practice is to configure swap at X 1.5 of the RAM size, but at system with a lot of memory (e.g. 128GB) can create swap with much less. It is always recommended to create some kind of swap.

Configure swap space:

1. Create a partition with Linux swap (82) type
2. mkswap to configure swap space, for example: mkswap /dev/vdb3
3. swapon to start using the swap space, for example: swapon /dev/vdb3  
   Can also be used to set priorities between different swap spaces.  
   Configure priority 10: swapon -p 10 /dev/vdb3  
   Display status: swapon -s  
   Higher priority means to be used before swap space with lower priority.
4. Add a line at /etc/fstab to maintain across reboots, advised to use UUID and doesn’t require a mount point as used by the OS directly:  
   UUID=”SWAP\_Space\_UUID” swap swap pri=10 0 0

## LVM

LVM allows to create RAID arrays between partitions.

1. Create partitions with disk type Linux LVM (8e).
2. Create a Physical Volume using: pvcreate /dev/PARTITION\_or\_DISK\_Name  
   Example: pvcreate /dev/vdc1 /dev/vdd  
   It still cannot be used by the OS and there is still no relationship between the reserved disks or partitions, but the disk space is reserved.  
   It is not a mandatory step, but is recommended to reserve the space.  
   pvs or pvdisplay - display PV’s
3. Create a Volume Group using vgcreate, e.g.: vgcreate vg0 /dev/sda2 /dev/sdb  
   Example: vgcreate vg1 /dev/vdd /dev/vdc1  
   Physical Extent are the smallest building blocks of the VG.  
   We can specify the size of the PE during the VG creation.  
   If not specified, the default PE size is 4MB.  
   vgs or vgdisplay - display VG’s
4. Create a Logical Volume: lvcreate -n LVNAME -L ###SIZE vg#  
   For example: lvcreate -n lv1 -L 100M vg1  
   Now the Logical Volume can be used by the OS layer.  
   The -l option allows to create LV using number of extents.  
   lvs or lvdisplay - display LV’s
5. Format the LV using mkfs.  
   Example: mkfs.ext4 /dev/vg1/lv1
6. A couple of options:
   1. Extend an existing VG in order to create a new LV:  
      Add /dev/sdc to VG1: vgextend VG1 /dev/sdc  
      Create a new lv: lvcreate -n BigLV -L 1G VG1  
      Format the new lv: mkfs.xfs /dev/vg1/BigLV
   2. Extend an existing VG in order to extend an existing LV:  
      Add /dev/sdc to VG1: vgextend vg1 /dev/sdc  
      Grow the lv: lvextend -L +1100M -r /dev/vg1/lv2  
      Note: if + is not mentioned, lvextend will resize the lv according to the size mentioned. Only if + is mentioned, it will grow it in accordance with the specified size.
   3. Replacing an old disk:  
      Add /dev/sdc to VG1: vgextend vg1 /dev/sdc  
      Free extents from /dev/sda2 using: pvmove /dev/sda2  
      Remove the old disk: vgreduce /dev/sda2
   4. Mount the lv:  
      Create mount point: mkdir /mnt/lv1fs  
      Mount: mount /dev/vg1/lv1 /mnt/lv1fs  
      Display mount point info: df /mnt/lv1fs  
      Filesystem 1K-blocks Used Available Use% Mounted on  
      /dev/mapper/vg1-lv1 95054 1550 86336 2% /mnt/lv1fs

Logical Volume structure example:

|  |  |
| --- | --- |
| File System level: mkfs | |
| lv1 - built from Logical Extents (there is an LE representation for each Physical Extent) | |
| vg1 | |
| pv level - /dev/vdd | pv level - /dev/vdc1 |

We can configure the Logical Extent to write each time to another disk in the group and this way maintain RAID0 or RAID1.

It is determined during the LV creation.

A Virtual device, e.g. VG or Software RAID, is mapped using a kernel mapper.  
This is what the /dev/mapper stands for.

In order to maintain consistency the /dev directory contains softlinks to those mapped devices.

For example, /dev/vg1/lv1 is a softlink to dm-2:

[root@clab1 lv1fs]# ll /dev/vg1/lv1

lrwxrwxrwx. 1 root root 7 Jan 27 17:37 /dev/vg1/lv1 -> ../dm-2

And dm-2 is actually /dev/mapper/vg1-lv1:

[root@clab1 lv1fs]# ll /dev/mapper/

total 0

lrwxrwxrwx. 1 root root 7 Jan 27 15:22 centos\_clab1-root -> ../dm-1

lrwxrwxrwx. 1 root root 7 Jan 27 15:22 centos\_clab1-swap -> ../dm-0

lrwxrwxrwx. 1 root root 7 Jan 27 17:37 vg1-lv1 -> ../dm-2

**lvm snapshots**

Create lvm snapshot using: lvcreate -s lv2snapshot -n SnapshotName -l #%ORIGIN\_SIZE

example: lvcreate -s lv1 -n lv1-snap -l 50%ORIGIN

When started, all logical extents on the snapshot are pointers to the original LE.

Once there is a change, the pointer is deleted, the original data is copied to the lv snapshot and the new data is written to the original lv.

If a snapshot is reaching its maximum size, the snapshot is corrupt.

# Display LV

[root@clab1 fs1]# lvs

LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert

root centos\_clab1 -wi-ao---- 7.57g

swap centos\_clab1 -wi-ao---- 924.00m

lv1 vg1 -wi-a----- 200.00m

# Create snapshot

[root@clab1 fs1]# lvcreate -s vg1/lv1 -n lv1-snap -l 50%ORIGIN

Logical volume "lv1-snap" created.

# Display LV

[root@clab1 fs1]# lvs

LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert

root centos\_clab1 -wi-ao---- 7.57g

swap centos\_clab1 -wi-ao---- 924.00m

lv1 vg1 owi-a-s--- 200.00m

lv1-snap vg1 swi-a-s--- 104.00m lv1 0.00

[root@clab1 fs1]# df -h

Filesystem Size Used Avail Use% Mounted on

devtmpfs 487M 0 487M 0% /dev

tmpfs 497M 0 497M 0% /dev/shm

tmpfs 497M 484K 496M 1% /run

tmpfs 497M 0 497M 0% /sys/fs/cgroup

/dev/mapper/centos\_clab1-root 7.6G 2.4G 5.2G 32% /

/dev/vda1 497M 205M 292M 42% /boot

tmpfs 100M 0 100M 0% /run/user/1000

tmpfs 100M 0 100M 0% /run/user/0

# Mount snapshot

[root@clab1 fs1]# mkdir /mnt/lv1-snap

[root@clab1 fs1]# mount /dev/vg1/lv1-snap /mnt/lv1-snap

[root@clab1 fs1]# cd /mnt/lv1-snap

[root@clab1 lv1-snap]# ls

lost+found test

# Corrupt the snapshot by filling it

[root@clab1 lv1-snap]# dd if=/dev/zero of=large bs=1M count=500

dd: error writing ‘large’: Read-only file system

137+0 records in

136+0 records out

143151104 bytes (143 MB) copied, 1.07046 s, 134 MB/s

## NFS

NFS server contains a file names /etc/exports listing all available shares.

NFS client needs the following services and packages to access NFS share:

* nfs-client
* nfs-utils, allow to run showmount -e NFSSERVER to display available shares.
* nfs-secure
* autofs

Security levels that can be assigned to a share:

* **none** - the owner of the files created will be nobody
* **sys** - based on the UID. For example, on the client PC the UID of Bob is 500, while on the server UID 500 is assigned to Eve. When using **sys** - Eve will be the owner of the files created. If no matching UID - the UID will be displayed.  
  From security perspective, if root on one machine will create a file on shares - it will be created and executed as root. To avoid this, root will fallback to **none** level and files created by him - will be owned by nobody.
* **krb5, krb5i, krb5p** - kerberos authentication, require kerberos server and kerberos key saved under /etc/krb5.keytab  
  The differences between krb version is the level of security:  
  krb5 - only encrypt the handshake  
  krb5i - also create checksum for the transferred data  
  krb5p - also encrypt all traffic, mount option: sec=krb5p

Mount NFS share example: mount -o sec=krb5p server:/exports/guests/ldapuser0 /home/guests/ldapuser0

enables to use auto mount, which means use NFS shares upon demand and not maintain a static connection.

2 options to configure:

1. At the file /etc/auto.master
2. Or under the directory /etc/auto.master.d/ edit the ldap.autofs file to contain the share root folder and the path to its configuration file.  
   For example: /home/guests /etc/auto.ldap  
   The configuration file contains destination subfolder, mount options and network path.  
   For example: ldapuser0 -rw,sec=krb5p server:/exports  
   To avoid configuring a different line for a different user, one can use & at the network path and it will take the \* username from the subfolder.  
   For example: \* -rw,sec=krb5p server:/exports/&

/etc/auto.master contains a configuration for /net which is a folder created upon demand based on the destination host, e.g. /net/clab2

In order to connect to SMB (CIFS) shares, need to specify the cifs type.

Example: mount -t cifs -o credentials=/dir/credentials\_file //server/.../share /mnt/winshare

The credentials\_file contains user & password in plain text, so verify only root has permissions on it.

To discover available shares on server execute: smbclient -L SERVERNAME

## Boot Process

**RHEL6 boot process**:

1. Power on
2. BIOS/EFI/UEFI
   1. POST (called BIST in the past)
   2. Boot loader - reading MBR to identify the GRUB (GRand Unified Bootloader) which executed the following steps:
      1. Stage 1 - manage the boot process until loading the Kernel, by reading stages 1.5 and 2.
      2. Stage 1.5 - provide the file system information of the /boot partition
      3. Stage 2 - GRUB interactive phase, allows to edit kernel parameters and to switch between operating systems.  
         Each entry at grub.conf has 3 lines:
         1. Hard disk location  
            Example: root (hd0)
         2. Which monolithic kernel to boot and with which parameters.   
            We are using a monolithic kernel to allow a minimal shared environment that can support many types of modules.  
            At this point /boot is the root of the operating system until the real OS is loaded, so the kernel location is set accordingly.  
            One of the main parameters that can be used is single, which loads the OS with minimum services and with root logged in without password. Allows to reset the root password.  
            Example: kernel /vmlinuz-3.10.0-229.14.1.el7.x86\_64 root=/dev/mapper/centos\_clab1-root ro LANG=en\_US.UTF-8 systemd.debug  
            GRUB can be secured by adding password to the file using MD5: password --md5 @$!$$T$TGdgd
         3. Since the kernel loaded is monolithic, we need to read the initial RAM disk which contains additional modules, not contained in the monolithic kernel. For example, one of the non-monolithic modules is LVM, so we cannot read from disk without it.  
            Example: initrd /initramfs-4.2.3-1.el7.elrepo.x86\_64.img  
            If prompt stuck on grub> it means GRUB failed to read grub.conf and GRUB waits for manual input.
   3. init (AKA Upstart) reading from /etc/inittab the runlevel to boot to. The scripts of each runlevel are under /etc/rc#.d . Use chkconfig to view all services and whether they are enabled on each runlevel:
      1. 0 - power off
      2. 1 - Single user (root only, no password), allows to reset root password
      3. 2 - Multi user (all users, minimal network services)
      4. 3 - 2 + full network services (Servers default)
      5. 4 - Undefined
      6. 5 - 3 + GUI (Desktops default)
      7. 6 - Reboot

The K/S at the start of the script line in the rc directory means Kill/Start respectively.  
rc directory example: [amit@clab1 etc]$ ll rc2.d/

total 0

lrwxrwxrwx. 1 root root 20 Oct 6 06:24 K50netconsole -> ../init.d/netconsole

lrwxrwxrwx. 1 root root 17 Jan 24 15:52 S10network -> ../init.d/network  
The means that netconsole will be stopped and network will start.

The problem with this boot process is that everything is dependent on scripts, which requires I/O, which is mostly a bottleneck. Also, the modules are loaded at a very late phase.c

**RHEL 7 Boot process**:

1. Power on
2. BIOS/EFI/UEFI
   1. POST (called BIST in the past)
   2. Boot loader - reading MBR to identify the GRUB (GRand Unified Bootloader) which executed the following steps:
      1. Stage 1 - manage the boot process until loading the Kernel, by reading stages 1.5 and 2.
      2. Stage 2 - GRUB2 has minimal instances of initrd, systemd and init.  
         Which means GRUB2 is a very minimal operating system.  
         Managed at /boot/grub2/grub.cfg and using it can specify modules to be available during the GRUB2 process.  
         This is why stage 1.5 is not required anymore, as GRUB2 can read /boot on its own.
         1. Different startup targets (rescue, emergency, multi user, graphical).   
            All targets require root password.  
            Set the required startup target using systemd.unit = TARGETNAME  
            Since all of the targets require root password, in order to reset root password - we need to break the GRUB2 process by appending rd.break to the kernel parameters line, which starts with linux16.  
            Example kernel line: linux16 /vmlinuz-4.2.3-1.el7.elrepo.x86\_64 root=/dev/mapper/centos\_clab1-root ro crashkernel=auto rd.lvm.lv=centos\_clab1/swap rd.lvm.lv=centos\_clab1/root rhgb quiet LANG=en\_US.UTF-8 systemd.debug  
            When using rd.break we are still using GRUB2, so the / of the filesystem is still /boot and the original / filesystem is loaded as /sysroot  
            Save changes using ctrl+X  
            The steps to reset root’s password:
            1. Reload /sysroot as RW instead of RO:  
               mount -oremount,rw /sysroot
            2. Chroot to /sysroot: chroot /sysroot
            3. Update root password:   
               echo NEWPASSWORD | passwd --stdin root
            4. When password is changed, /etc/shadow creates a new instance of itself. If SELinux is enabled, it is not working at rd.break state so we need to update .autorelabel . This allows the new /etc/shadow to take effect after boot, with SELinux permitting it: touch /.auotrelabel
3. systemd boots services. init is available for backward compatibility.  
   systemd manages units:
   1. Services
   2. Sockets
   3. Mounts
   4. Targets are checkpoints between every phase, whether the current target is sufficient (AKA isolated) or that additional services are required to boot. Available targets:
      1. 0 - power off
      2. 1 - Rescue / Emergency (system starts in RO)
      3. 2,3,4 - Multi user
      4. 5 - Graphical
      5. 6 - Reboot

The target is set at /etc/systemd/system/default.target

The .../default.target file is a softlink to the matching directory.

Example: [amit@clab1 ~]$ ll /etc/systemd/system/default.target

lrwxrwxrwx. 1 root root 37 Oct 6 06:29 /etc/systemd/system/default.target -> /lib/systemd/system/multi-user.target

Display default target: systemctl get-default

Change default target: systemctl set-default REQUIREDTARGET

Display the current target dependencies: [amit@clab1 ~]$ systemctl list-dependencies | grep target

Example output:

default.target

● ├─basic.target

● │ ├─paths.target

● │ ├─slices.target

● │ ├─sockets.target

● │ ├─sysinit.target

● │ │ ├─cryptsetup.target

● │ │ ├─local-fs.target

● │ │ └─swap.target

● │ └─timers.target

● ├─getty.target

● ├─nfs-client.target

● │ └─remote-fs-pre.target

● └─remote-fs.target

● └─nfs-client.target

● └─remote-fs-pre.target

Change between targets: systemctl isolate TARGETNAME

For example: systemctl isolate multi-user.target

For the isolate option to be available for the specific target, we need to set allowisolate=yes in the target conf file.

Masking a target disabled the option of starting a unit by mistake.

grub2-mkconfig is used to fix the grub menu entries, by re-reading the available target files.

If there has been a change in the target files, it will not fix them.

Update grub2-mkconfig to grub configuration: grub2-mkconfig > /boot/grub2/grub.cfg

In order to edit the GRUB menu:

1. Edit /etc/default/grub file with the required parameters
2. Verify at grub2-mkconfig output
3. Write changes: grub2-mkconfig > /boot/grub2/grub.cfg

In order to change the default boot entry use: grub2-set-default

## Firewalld

netfilter is a module that enables to develop firewall applications.

Until RHEL7 it was only used by iptables.

Starting from RHEL7 firewalld is also using the netfilter module.

firewalld and iptables cannot co-exist on the same system.

firewalld is simpler, but does not support advanced rules, for examples use of regex.

firewalld is using zones to filter the traffic.  
Zones are set per connections.

firewalld can be managed using cli or GUI.

CLI command: firewall-cmd

Display default zone:

[amit@clab1 ~]$ firewall-cmd --get-default-zone

public

Display rules that apply to a zone, e.g. public:

[amit@clab1 ~]$ firewall-cmd --list-all --zone=public

public (default, active)

interfaces: eth0

sources:

services: dhcpv6-client ssh

ports:

masquerade: no

forward-ports:

icmp-blocks:

rich rules:

masquerade allows to NAT the connection behind a different IP.

To maintain changes across reboots use the --permanent switch.

Example to add support for port 8080 on the public zone: firewall-cmd --permanent --add-port=8080/tcp --zone=public

In order to update a new policy execute: firewall-cmd --reload

Can also run the command without the --permanent switch, for example add listening on the DNS service:

[amit@clab1 ~]$ sudo firewall-cmd --add-service=dns --permanent

And then update the current firewall by running again without the --permanent will update on the fly for the current boot:

[amit@clab1 ~]$ sudo firewall-cmd --add-service=dns

For example, discover which ports will the firewall open when adding the service dns:

Search for the dns configuration file used by the firewall:

[amit@localhost ~]$ rpm -ql firewalld | grep dns

/usr/lib/firewalld/services/dns.xml

/usr/lib/firewalld/services/mdns.xml

Display dns.cml file to know which ports it allowed:

[amit@localhost ~]$ sudo cat /usr/lib/firewalld/services/dns.xml

<?xml version="1.0" encoding="utf-8"?>

<service>

<short>DNS</short>

<description>The Domain Name System (DNS) is used to provide and request host and domain names. Enable this option, if you plan to provide a domain name service (e.g. with bind).</description>

<port protocol="tcp" port="53"/>

<port protocol="udp" port="53"/>

</service>

GUI to manage firewalld: firewall-config

Note to choose **permanent** from the drop down to maintain changes across reboots.

**Lockdown** blocks all traffic except whitelisted services.

**Panic mode** blocks all traffic.

# RH254

Instructor: Amir Sela [amirse@gmail.com](mailto:amirse@gmail.com)

## Systemd

The main different between systemd and the older init/upstart is that systemd can start services in parallel, while init/upstart started them sequentially.

List systemctl services dependencies: systemctl list-dependencies

systemd searches for unit (.service .socket etc…) files in specific folders and loads them:

/usr/lib/systemd/system is commonly used to hold the unit files for the services that came with the system.

/etc/systemd/system is commonly used for user created unit files.

systemd can also do other tasks: automount when specific user does something, has internal cron etc…

Example for service status: [amit@clab1 ~]$ systemctl status sshd

● sshd.service - OpenSSH server daemon

Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)

Active: active (running) since Sun 2016-02-14 15:26:08 UTC; 36min ago

Main PID: 1323 (sshd)

CGroup: /system.slice/sshd.service

└─1323 /usr/sbin/sshd -D

Feb 14 15:26:08 clab1.amitlab systemd[1]: Started OpenSSH server daemon.

Feb 14 15:26:08 clab1.amitlab systemd[1]: Starting OpenSSH server daemon...

Feb 14 15:26:09 clab1.amitlab sshd[1323]: Server listening on 0.0.0.0 port 22.

Feb 14 15:26:09 clab1.amitlab sshd[1323]: Server listening on :: port 22.

Feb 14 15:26:46 clab1.amitlab sshd[2443]: Address 192.168.122.1 maps to gateway, but this does not map back to the address - POSSIBLE BREAK-IN ATTEMPT!

Feb 14 15:26:47 clab1.amitlab sshd[2443]: Accepted publickey for amit from 192.168.122.1 port 34289 ssh2: RSA a1:ac:9a:b3:39:48:64:36:b0:b7:29:cf:47:c3:53:ba

loaded means the file was loaded to the RAM.

Status is the active indication.

Service file example: [amit@clab1 ~]$ cat /usr/lib/systemd/system/sshd.service

[Unit]

Description=OpenSSH server daemon

After=network.target sshd-keygen.service

Wants=sshd-keygen.service

[Service]

EnvironmentFile=/etc/sysconfig/sshd

ExecStart=/usr/sbin/sshd -D $OPTIONS

ExecReload=/bin/kill -HUP $MAINPID

KillMode=process

Restart=on-failure

RestartSec=42s

[Install]

WantedBy=multi-user.target

At the section [Service] can configure what to load before the service and the command line to load.

ExecReload is used to re-read the configuration files.

[Install] section tells at which targets to load.

View boot process log under /var/log/boot.log

Disable service from loading automatically: systemctl disable SERVICENAME

Totally disabling the service: systemctl mask SERVICENAME

View service status with 1 result for scripts: systemctl is-active SERVICENAME

systemctl actions starting with cond… allow conditional actions, for example condrestart for HTTPD when no one is connected.

## Boot process

BIOS -> Bootloader (grub2) -> Kernel + initrd -> systemd

In *emergency* the filesystem is read only, commonly used to run fsck on the file system.

In *rescue* the filesystem is in read-write mode.

In order to recover the root password edit the boot parameters to boot to *emergency* mode:

* (if exists) Remove the configuration to write to serial console=ttyS0,115200n8 so that the output will be written to the screen and not to COM1 (LEAVE console=ttyS0 in place)
* Append rd.break at the end of the linux16 line

Reset root password in *emergency* mode:

Mount the file system: mount -o remount,rw /sysroot

Change to root: chroot /sysroot

Change root password: passwd

Touch the /.autorelabel file for SElinux to update its configuration: touch /.autorelabel

## IPv4

Add ip address: ip addr add IP/SubnetMask dev DEVICE\_NAME

Example: ip addr add 172.25.0.120/24 dev eth0

To allow ifconfig to be aware of it add backward compatible label, e.g.:

ip addr add 172.25.0.120/24 dev eth0 label eth0:0

Network Manager still use files at /etc/sysconfig/network-scripts/ but they now represent a configuration file and not a device.

Edit connection using: nmcli con mod or nmcli con edit CONNECTION\_NAME

At the edit mode can get help by using describe PARAMETER

## IPv6

Discover IPv6 address using dig -t AAAA

For example on [www.facebook.com](http://www.facebook.com): dig -t AAAA www.facebook.com

IPv6 has 128 bits and uses hexadecimal to represent addresses.

Hexadecimal representation: 0-9 A(10) B(11) C(12) D(13) E(14) F(15)

Example IPv6 address: 2a03:2880:f01a:1e:face:b00c:0:25de

There are 8 octets, each containing 4 characters.

If a character is missing - it means there is a 0 there.

In a valid IPv6 address, only one abbreviation of :: which translates to 0’s can exist. Therefore, 2001:db8:0:7::2 is valid while 2001:db8::7::2 is invalid since there are 2 abbreviations.

Also works in IPv4, so ping 127.1 translates to 127.0.01

Add an IPv6 IP to an existing connection, e.g. to eth0 device::

nmcli con mod eth0 ipv6.method manual ipv6.addresses 2000::aaa1/64

Examples for nmcli usage: man nmcli examples

Restart the connection: nmcli con down eth0; nmcli con up eth0

Ping IPv6: ping6

When pinging local address need to specify interface through which to execute the ping: ping6 -I eth0 2000::aaa2

Or:

ping6 2000::aaa2%eth0

## Link Aggregation

In order to aggregate 2 and more physical network cards set the devices as slave and create a connection master.

Red Hat also name it **teaming**.

Previously it was done using a driver named *bonding*.

Starting from RHEL7 it is accomplished using a daemon called *Teamd*.

Command to control *Teamd* is teamdctl.

1. Added 2 network cards: ens9 and ens10
2. Display all cards, regardless of status: ip link
3. Add new connection type team. Teamd uses a runner configuration written in Json:  
   nmcli con add type team con-name TEAM0 ifname team0 config '{ "runner": { "name":"activebackup" } }'  
   Note: The default configuration is round robin.  
   View Json examples under the EXAMPLES section in man teamd.conf
4. Setup IP: nmcli con mod TEAM0 ipv4.addresses 192.168.122.169/24 ipv4.method manual
5. Add slave devices:  
   Add ens9: nmcli con add type team-slave ifname ens9 con-name TEAM0-slave0-ens9 master team0  
   Add ens10: nmcli con add type team-slave ifname ens10 con-name TEAM0-slave0-ens10 master team0
6. Start team: nmcli con up TEAM0
7. View team state: teamdctl TEAM0 state
8. Test teaming by aggressively taking down one of the links: ip link set ens9 down  
   Another method: nmcli con down TEAM0-slave0-ens9  
   Another method: nmcli dev dis TEAM0-slave0-ens9
9. Start back downed device: ip link set ens10 down  
   Another method: nmcli con up TEAM0-slave0-ens9

Tell NetworkManager to reload updated configuration from scripts files, e.g. from TEAM0 file: nmcli con load /etc/sysconfig/network-scripts/ifcfg-TEAM0

teamnl is used to display options available for teamed device.

## Bridging

Connect 2 different ethernet networks through the same device in a layer 2 level, even without an IP.

It is created similarly to teaming, only the type is bridge which uses bridge-slave and no need for runner and IP configuration.

It is used to listen to network traffic silently.

brctl show is used to display bridging information.

Firewall

Zones can be assigned per interface.

Default zone is public, see: firewall-cmd --get-default-zone

List all zones and their configuration: firewall-cmd --list-all

Add allow rule for a well known service: firewall-cmd --add-service=http --permanent

Add specific port: firewall-cmd --add-port=8080/tcp --permanent

ncat is a utility that listens on traffic per given port, structure: ncat -l Port#

For example, listen on port 80: ncat -l 80

Can also connect to remote server using ncat, e.g. connect to server 1 on port 80: ncat server1 80

Zones configuration firewall are saved under: /etc/firewalld/zones/ZONENAME.xml

In order to create rich rules, which allow advanced filtering, use --add-rich-rule.

Manual and examples: man firewalld.richlanguage

For example, allow access to IMAPS service only from the address 1.1.11 permanently: firewall-cmd --add-rich-rule='rule family=ipv4 service name=imaps source address=1.1.1.1 accept' --permanent

Example to add access to port 12345 on TCP only from IP 1.1.1.1: firewall-cmd --add-rich-rule='rule family=ipv4 port port=12345 protocol=tcp source address=1.1.1.1 accept' --permanent

Both Masquerading/SNAT and Port forwarding are instances of NAT.

The regular NAT is named SNAT in Linux, since it changes the source address.

Port forward is named DNAT in Linux, since we change the destination address.

Examples:

Configure SNAT for all traffic routed through the home zone:   
firewall-cmd --zone=home --add-masquerade

Configure DNAT :

firewall-cmd --add-rich-rule='rule family="ipv6" source address="1:2:3:4:6::" forward-port to-addr="1::2:3:4:7" to-port="4012" protocol="tcp" port="4011"

ss is used to list network addresses and ports in use.

## DNS

Recursive query is done from Client to Server and means either provide an answer or not.

Iterative query is done from Server to Server and means either provide an answer or direct to another server that might know the answer.

The root servers (.) know one level below (.com. or .il. etc...) and redirect accordingly.

Those root servers are preconfigured in each DNS server flavor.

DNS servers are updated in /etc/resolv.conf

DNS are divided into Resource Records (RR):

|  |  |  |
| --- | --- | --- |
| **Name** | **Query** | **Reply** |
| A - Address record | Hostname | IP |
| PTR - pointer | IP in the domain in-addr.arpa  Example: query the IP 212.179.2.1 will be done using 1.2.179.212.in-addr.arpa | Hostname |
| CNAME - Canonical name | Hostname  Example: [amit@clab1 ~]$ host www.nba.com  www.nba.com is an alias for www.nba.com.edgesuite.net.  www.nba.com.edgesuite.net is an alias for a1570.gd.akamai.net.  a1570.gd.akamai.net has address 212.199.202.68  a1570.gd.akamai.net has address 212.199.202.44 | Hostname |
| AAAA - ipv6 address | Hostname | IPv6 |
| MX - mail exchange | Mail Domain | Hostname of mail servers Priorities are processed from lowest to highest (lowest is preferred) |
| NS - name server | Domain name | Hostname of DNS server |

**Caching Name server** is used to cache DNS results for better performance based on their TTL record.

bind is a DNS server that has been used for years.

Starting from RHEL7 the recommended DNS server is unbound

There are a couple of tools to query DNS in Linux: dig, nslookup, host, getent

dig is using only DNS to query, not even the local hosts file.

getent is used to lookup not only DNS and will also query the local hosts file. It is configured at /etc/nsswitch.conf which contains lists of databases to query depends on the query.

For example, hosts query will check at files (/etc/hosts), then lookup the dns server:

hosts: files dns

Example query: getent hosts www.nba.com

It is the default client used by the operating system.

Example for the databases to use when querying for a user:

passwd: files sss

Example query: getent passwd amit

When using the passwd switch it will search for files first and then at sss

[amit@localhost ~]$ netstat -lnpt

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name

tcp 0 0 127.0.0.1:5900 0.0.0.0:\* LISTEN -

tcp 0 0 127.0.0.1:5901 0.0.0.0:\* LISTEN -

tcp 0 0 0.0.0.0:111 0.0.0.0:\* LISTEN -

tcp 0 0 127.0.0.1:5939 0.0.0.0:\* LISTEN -

tcp 0 0 192.168.122.1:53 0.0.0.0:\* LISTEN -

tcp 0 0 0.0.0.0:22 0.0.0.0:\* LISTEN -

tcp 0 0 127.0.0.1:631 0.0.0.0:\* LISTEN -

tcp 0 0 127.0.0.1:25 0.0.0.0:\* LISTEN -

tcp6 0 0 :::111 :::\* LISTEN -

tcp6 0 0 :::22 :::\* LISTEN -

tcp6 0 0 ::1:631 :::\* LISTEN -

tcp6 0 0 ::1:25 :::\* LISTEN -

netstat provides information about current network connections. Main switches:

l - list only ports in LISTEN mode

n - numerical info

p - list the listening port

t - list only tcp

u - list only udp

unbound configuration file is /etc/unbound/unbound.conf

After installing unbound, note to:

1. Enable unbound to start automatically: systemctl enable unbound
2. Start the unbound service: systemctl start unbound
3. Allow the service on the firewall: firewall-cmd --add-service=dns --permanent
4. Configure the interface to listen on at /etc/unbound/unbound.conf
5. Configure the clients allowed to execute recursive queries at /etc/unbound/unbound.conf

Check /etc/unbound/unbound.conf for syntax errors: unbound-checkconf

Manage unbound using: unbound-control

DNS Forwarder means that the DNS server will act as a client against another DNS server. For example, query the ISP servers for faster response than the root servers.

Forwarder can be done for all addresses (.) or for specified domain.

trust-anchor is used to validate entries and queries.

One can also disable validation by configuring the domain-insecure on the local domain.

Use dig to query specific type, for example MX: dig -t mx telemessage.com

## Configuring Email Transmission

The default mail service is postfix.

In the past sendmail was the default.

SMTP server is also called MTA (Mail Transport Agent).

Emails reaching the system are saved under /var/spool/mail

List all email using: mail

**Main postfix configurations**

The postfix conf file is /etc/postfix/main.cf

It can also be edited using postconf

Outgoing settings:

relayhost = [IP] - mail relay server to route emails through

myorigin = domainname.com - email domain address to send from

Incoming settings:

inet\_interfaces - Interfaces to listen on  
To listen on all interfaces: inet\_interfaces = all

mydestination - local domains to receive email for

mynetworks - networks from which relaying is allowed

local\_transport - how to deliver email addressed to $mydestination

Email aliases of all mail services (postfix, sendmail etc..) are configured at the file /etc/aliases

Update aliases using newaliases

## Remote Storage

Scsi controllers come in Hardware and Software flavors.

Today Hardware flavors are no longer used.

iScsi allows the local network card to manage remote disks (block devices).

Terminology:

**initiator** - iScsi client

**target** - iScsi storage resource

**ACL** - list of allowed IQN

**discovery** - querying server for configured targets

**IQN** - iScsi qualified name, a worldwide unique name in the format:

iqn.YYYY-MM.com.reversed.domain[:optional\_string]

**LUN** - Logical Unit Number

**portal** - an IP address and port on target or initiator that is used to establish connections

**TPG** - Target Portal Group, the IP:Port set on the storage server

iScsi service or the storage resource is target

targetcli is used to manage iScsi targets on Storage server.

/etc/services contains all the ports that well known services are using.

Setup iScsi target:

1. Allow Firewall access to port 3260/tcp
2. Login to targetcli
   1. Commands in targetcli appear under their respective hierarchy, for example create fileio target under /backstores/fileio create
   2. ls is also supported to display what is available at each level.
3. Create target block or file device. For example, create file (/var/storage0) device sized 1G named storage0:/backstores/fileio create storage0 /var/storage0 1G
4. Create WWN, example: /iscsi create iqn.2016-02.com.amit.clab2:storage0
5. Configure ACL, example:   
   iqn.2016-02.com.example.server0/tpg0/acls create iqn.2016-02.com.example.desktop0
6. Create LUN, example:   
   iqn.2016-02.com.example.server0/tpg0/luns create /backstores/fileio/storage0
7. Create Target Portal, example:   
   iqn.2016-02.com.example.server0/tpg0/portals create (will create on all IP’s with the default port 3260)
8. Start the target service

The actual LUN file is a sparse file, similarly to thin provision. When using ls it will display the maximum size the file can be, but in order to see the actual disk usage run du -sh on it

When checking the status of the target status it will appear as active (exited) since it runs a script which updates kernel parameters for the iScsi setup and then exits.

Setup iScsi initiator:

1. the service iscsi is used to manage iscsi initiators - requires a configuration to start successfully
2. Configure initiator IQN at /etc/iscsi/initiatorname.iscsi
3. Manage iscsi initiator using iscsiadm
4. Discover targets on server:   
   iscsiadm --mode discoverydb --type sendtargets --portal iScsiServerName --discover
5. Login to a discovered target: iscsiadm --mode node --targetname iqn.2016-02.com.amit.clab2:storage0 --portal iScsiServerName --login
6. Execute dmesg to see the adding of the new disk
7. List all block devices: lsblk
8. Then the iScsi disk will appear as any other disk and can be fdisk’d
9. When rebooting for the 1st time after configuring iScsi disks - there might be a bug to shutdown the server, so after all service stop - just shut it manually
10. In order to mount it during boot, need to configure it as iScsi using the \_netdev option at the /etc/fstab so it will know to wait for the network before loading it:   
    UUID=b9c1fcdd-6be8-4d3e-a9de-4bc2f279c744 /MountDir xfs \_netdev 0 0

## NFS

NFS supports RPC (Remote Procedure Call): the client sends a function (*open(), close(), write(), read()*) to the NFS server and the server executes it.

Until NFS3 including each function required a different random port.

The client queried the portmap service for the port to access for each function.

Starting from RHEL7 the NFS version is 4yi.

Starting from NFS4 there is better security features, such as kerberos support, tcp based by default instead of udp, use of one static port for each function etc…

For NFS4 need only to allow access on the port 2049/tcp.

In order to create an NFS share, need to install nfs-utils.

The NFS shares are managed on the server using /etc/exports.

Share structure:

Local Path To Share Allowed IP’s(Export Options)

/hello \*.example.com(sync,ro)

/hello 192.168.13.0/24(async,rw)

If there is no space between the Allowed IP’s and the export option, it will translate it differently. For example: /hello 192.168.13.0/24 (async,rw) will translate to:   
/hello 192.168.13.0/24(defaults) \*(async,rw)

defaults export options stand for read-only.

The sync/async option sets how to notify about write changes to disk: when to notify the client that it completed writing to disk, only once completed (sync) or already when part of the file completed (async), which allows faster writing.

The nfs server service is called nfs-server.

Sames as iScsi, it is not a real service, but a script that updates the kernel with the shares to allow and then exits. This is why it appears started(exited).

On the nfs client:

1. Create directory to mount to: mkdir /mnt/hello
2. Mount the nfs share: mount servername:/hello /mnt/hello  
   No need to mention -t nfs since it auto detects the destination protocol.
3. Configure mount at /etc/fstab to maintain access across reboots using the nfs file system type.

In nfs 2 and 3 one could query the available exports on destination server:

showmount -e destserver

In nfs 4 one can display the available exports by mounting to / on the destination server.

The mount is carried out using the root user.

In nfs 2 and 3 file operations are done by the client telling the server which user and UID is doing the operation and the server allowing it based on the same local UID.

For example, Bob with UID 300 at the nfs client - will be allowed access based on the permissions the user alice with UID 300 has on the nfs server.

Any operation done from the client to the server using the root user is automatically degraded to be done using the user nfsnobody.

This root squashing operation is done to disallow root access to destination server.

To disable it add the option no\_root\_squash at the mount export options.

By default, an nfs server will not allow write operations in the first 90 seconds to complete operations leftovers from before the restart.

When an NFS export supports kerberos, it means that the client is required to provide a kerberos ticket from a domain controller to get access to the share.

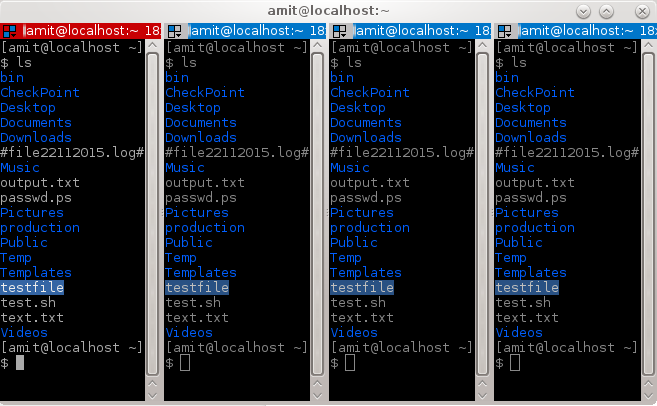
Also, need to start the service nfs-secure-server on the server side.  
Now, after mounting the export we will need to access files while authenticated to the domain.

Starting from nfs 4.2 SELinux is also supported, so that the SELinux context is verified as well.

terminator is GUI utility that allows to duplicate the command over all open consoles.

In order to open additional console, open terminator and then right-click and choose to split vertically or horizontally.

In order to broadcast commands, right-click the menu button and choose broadcast all.



## Samba

Allows access and serve CIFS shares, so that Windows servers can access it.

Uses port 445/tcp by default and for backward compatibility also the ports 139/tcp, 138/udp and 137/udp.

Configure CIFS shares using /etc/samba/smb.conf

**Setup Samba server**

1. Install the samba package: yum install samba  
   This will install 2 services: nmb for NetBIOS and smb for the Samba
2. Configure permissions on the directory to be shared:
   1. Create dir: mkdir /docs
   2. Configure permissions: chmod 777 /docs
   3. Configure SELinux: semanage fcontext -a -t samba\_share\_t “/docs(/.\*)?”  
      Apply the new SELinux settings: restorecon -RFvv /docs  
      Preferably not to use chcon, because everything not configured using the SELinux configuration will be lost during relabel. A relabel operation is running when the system boots into rescue mode, when SELinux is disabled, which results in an SELinux relabel when the system boots normally again.
3. Configure /etc/samba/smb.conf  
   The only sections that are of interest are General and the Shares configuration.
4. Setup Linux users with NTLMv2 passwords:
   1. Create user on Samba server, do not configure password so it cannot connect using Linux and disable shell access
   2. Create NTLMv2 password compatible with Samba:   
      smbpasswd -a USERNAME
5. Start Samba service and allow access on the firewall:  
   systemctl enable samba  
   systemctl start samba  
   firewall-cmd --add-service=samba --permanent
6. Verify client access   
   List remote Samba shares as Remote user:   
   smbclient -L SERVERNAME -U RemoteServerUsername  
   List remote Samba shares without entering a username:   
   smbclient -L SERVERNAME -> Click Enter when prompted for password for anonymous login  
   Connect to remote Samba share: smbclient //ServerName/ShareName -U USERNAME  
   To create a permanent connection mount using /etc/fstab

pdbedit is used to manage the Samba users.

Display configured Samba users: pdbedit -L

Home directories is allowed by default, see /etc/samba/smb.conf section:

[homes]

comment = Home Directories

browseable = no

writable = yes

But it doesn’t work since we need to enable SELinux Boolean:

setsebool -P samba\_enable\_home\_dirs on

cifscreds allows adding alternate credentials for Samba shares.

## MySQL / MariaDB

MariaDB is a fork of MySQL.

It is a relational DB.

1. Search for yum group, including those hidden by default: yum grouplist hidden
2. Install MariaDB: yum groupinstall mariadb mariadb-client
3. Start MariaDB service
4. Execute MariaDB 1st time wizard: mysql\_secure\_installation  
   Use the secure option to secure the DB, which comes open by default
5. Connect to MariaDB interactively: mysql -p
6. List databases: show databases;
7. Create Database named store: create database store;
8. Help file, for example on create database: help create database;
9. Change to database store: use store;
10. Connect directory to store: mysql -p store
11. Create table named products with keys: create table products (id int(6) auto\_increment primary key, name varchar(20) not null, description varchar(100), instock boolean default 0, price int(6) not null);
12. View table description: describe products;
13. 4 main operations: Create (insert), Read (select), Update (update), Delete (delete)  
    Note: doesn’t require commit after adding or changing entries.
14. Insert new product: insert into products (name,description,instock,price) values ('Razer mouse', 'Gaming mouse',1,150);
15. Display all info from the products table: select \* from products;
16. Create User: create user ‘USERNAME’@’localhost’ identified by ‘PASSWORD’;
17. Add permissions: grant update,insert,select,delete on store.products to ‘username’@’localhost’;
18. Display user permissions: show grants for ‘username’@’localhost’;
19. Remove permissions using revoke

MySQL conf file: /etc/my.cnf

Can change that connect are only allowed from 127.0.0.1

Hot backup (during runtime) will be done using mysqldump -u USERNAME -p --all-databases >file.dump

This will write all DB to a text file with all commands to restore the DB to the current state.

Restore: mysql -p <file.dump

Cold backup is done by stopping the mariadb service and copying the actual db files from /var/lib/mysql

## Apache

Service name is httpd.

The standard HTTP command browsers are using is GET.

For example, retrieve picture.jpg from server0.example.com using:

GET /picture.jpg HTTP/1.1

HOST:server0.example.com

When using HTTP1.0 no need to specify host name.

When browsing to the web server without specifying the specific page - it will return its default page (default: index.html).

The default installation contains only the basis modules, even the HTTPS (package name: mod\_ssl) module should be added.

In order to harden the Apache security - remove unused modules.

The important directories Apache uses are:

* /var/www
  + /cgi-bin/ -
  + /html/ - The actual site pages
* /etc/httpd
  + /conf/httpd.conf - main Apache conf file read by Apache during service start
  + /conf.d/\*.conf -
  + /conf.modules.d/\*.conf -

When using SELinux - need to update it as well when changing the default listen address.

Log directory: /var/log/httpd

All <Directory> sections contain security related information.

For example, the Apache security settings for the operating system root directory is:

<Directory />

AllowOverride none

Require all denied

</Directory>

Virtual hosts allows serving different sites using the same Apache web service. The route decision is done based on the submitted URL or port.

When creating new subdirectories for sites not under the default /var/www/html/ - remember to apply the same SELinux context as the one applied on the default directory httpd\_sys\_content\_t.

Also, remember to adjust the <Directory> section for the new location in the httpd.conf.

Example for VirtualHost, so that any request to the URL abc.example.com is serviced using the /sites/abc/html directory:

<VirtualHost \*:80>

ServerName abc.example.com

DocumentRoot /sites/abc/html

</VirtualHost>

When using VirtualHost also need to configure the primary site, which worked by default before, as VirtualHost as well.

**SSL** is a negotiation protocol which allows to agree which encryption method to use.

There are 2 types of encryption: Symmetric and Asymmetric.

Symmetric means both ends use the same key and certificate to encrypt the data.

Asymmetric means that each side is using different keys: the client uses the public key to encrypt the data, but only using the private key it can be decrypted.

In order to validate the site authenticity, public certificates should be signed by a Certificate Authority.

Cipher is the encryption algorithm.

The key is with which key will the data be encrypted.

The crypto-utils package contains the genkey utility to create private SSL keys.

The mod\_ssl package is configured using /etc/httpd/conf.d/ssl.conf

The key and public certificates for the primary site is configured at /etc/httpd/conf.d/ssl.conf while the pair for the other Virtual hosts is configured at /etc/httpd/conf/httpd.conf.

Apache manual can be obtained by installing httpd-manual and available through <http://servername/manual>

Behind most servers there’s a web application, AKA CGI, which generates personalized/dynamic pages per logged in user behind the same url.

**CGI** stands for *Common Gateway Interface*.

Basically, the Apache queries the CGI application, which creates the unique page and returns it to the Apache server.

Apache uses predefined scripts to query the cgi application.

Those scripts should be located in /var/www/cgi-bin.

In order for Apache to access it:

1. Create a script alias from /var/www/html/cgi-bin to /var/www/cgi-bin at the httpd.conf file: ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"
2. Allow the cgi SELinux boolean

CGI even supports queries in BASH.

Today it is more common to use queries inside the web pages using PHP or Python (DJango/WSGI).

It is configured at httpd.conf using WSGIScriptAlias parameter.

## Bash scripting

Escape characters:

\ is used to escape only the character after it. E.g. \$ [] will escape only $

"" is used to escape spaces and wildcards. E.g.   
Will not escape variables:

amit@clab1 /v/w/html> echo "I have $5"

Output: I have

Will escape space:

amit@clab1 /v/w/html> echo "3 4 5"

Output: 3 4 5

'' is used stronger than ""

When commands are interpreted, bash starts interpreting and then passes to the command.

Therefore, the output of find / -name \*.txt and find / -name “\*.txt” is different.

This is because bash interprets \* differently than find.

It will look in the current directory for any file ending with .txt, but when passing “\*.txt” - bash doesn’t interpret the command and passes it to find, which uses it as wildcard.

For example:

We have the file 1.txt in the folder:

amit@clab1 /tmp> ls

1.txt

When running the following command:

amit@clab1 /tmp> sudo find / -name \*.txt | wc -l

bash interprets it as:

amit@clab1 /tmp> sudo find / -name 1.txt | wc -l

And then the output is:

1

But when running the following command:

amit@clab1 /tmp> sudo find / -name '\*.txt' | wc -l

Bash passes '\*.txt' as is to find and therefore the output is:

288

`` is equivalent to $()

So `date` will yield the same output as $(date):

[amit@localhost ~]$ echo `date`

Sun Mar 13 18:35:35 IST 2016

[amit@localhost ~]$ echo $(date)

Sun Mar 13 18:35:46 IST 2016

To configure system wide variables use export or add to /etc/profile

Loop to go line by line in a file:

While read line

do

Echo $line

done < 1.txt

## 

## 

## April 3, 2016 Exercise

Questions:

\* You're supposed to use clean serverX and desktopX machines for this - please run rht-vmctl reset on them first

\* everything you do in this lab \*MUST\* persist across reboots

\* If not stated, do it on serverX. It will be stated when to use desktopX

Part 1 - users, user environment and scripts

1) Create 40 users named user1-40. The user IDs should begin with 3001, such that user1 has the UID 3001, user2 has the UID 3002, and so on. Every user should have a personal bin directory automatically created in their home directory (for example, the user user34 should have a directory called /home/user34/bin)

2) Configure every user's password to be their own username

3) Configure the accounts so that the first time a user logs in, s/he must change the password

4) Set a system-wide bash alias such that a user can run the command "findpdfs", and it will list all the pdfs in the directory he's running it (and below)

5) Configure a system-wide environment variable such that every user on the system can type 'echo $color' on the command prompt, and gets back 'green' (without the quotes of course)

6) modify the system-wide PATH variable so that every user has the directory ~/bin in their PATH (test this and see that it works, which means - that every user can create a script in their personal bin directory and run it from everywhere without having to type the entire path or ./script)

7) user17 needs her home directory backed up each day into a tar file in /tmp, but she doesn't know how to use cron. Create a cronjob for her to backup her home directory at 2am every day into a tar file in /tmp called user17\_home.tar.gz

8) user36 keeps deleting his file "shopping.txt" (create that file in his home dir). Create a hardlink to that file so you can easily restore it to him every time he deletes it

9) user12 always goes into the directory /usr/share/doc/vim-common-7.4.160/. Create a symlink in his home folder called vimdocs that takes him directly to that path

10) users1-10 need an extra bash alias (but only them), that runs the command "chmod 777" whenever they type "full\_access".

11) Create a group called sales, and add user1-10 to it as their secondary group

12) Create a group called developers, and add users11-20 to it as their secondary group

13) Create a directory called /home/sales. This directory should be owned by user1, with its group being sales. The permissions for the folder should be 2770

14) Add an ACL to /home/sales so that user37 gets read/cd access to everything

15) Make sure that every single file or directory created in /home/sales gets a proper ACL to let user37 access it

16) user14 is an outside contractor and will be leaving in 1 year. Set her account to expire one year from now

Part 2 - storage management

1) Create a gpt disk label on /dev/sdb

2) Create 3 partitions, each sized 2GB, on that disk

3) Partition 1 will be an LVM physical volume. Partition 2 will also be an LVM physical volume. Partition 3 will be a regular XFS partition

4) Format partition 3, and make sure it gets mounted on boot under /storage

5) Create an LVM VG out of the two LVM partition. the VG name will be vg00

6) Create a new LV inside vg00 called "docs". Its size will be 512M, using 16M Physical Extent size

7) Use the target service(iscsi service) to share out the docs LV to desktop0

8) Use NFS to export out /storage (from step 4) to desktop0 (use your local LAN ip address range on the export line)

9) On desktop0: use the iSCSI client tools to connect to the shared LV on server0. Once you managed to do that, create a GPT label on the LUN, create 1 partition from all its space, format the partition using ext4, and mount it under /docs (And of course, this should happen after a reboot as well)

10) On desktop0: mount the server0 exported /storage on your machine, also on /storage (and make sure it mounts across reboots)

Part 3 - Networking

1) Add two new NICS to both server0 and desktop0. All four NICS should be in a separate virtual LAN that only they share (in VMWare workstation, for example, create them on vmnet7 or something like that)

2) Configure teaming on both desktop0 and server0 from the pairs of devices

3) Set static ips for the links. desktop0 will be 192.168.0.1/24, and server0 will be 192.168.0.2/24.

4) Check that the teaming works by pinging across the link, and dropping the active NIC

Part 1 - Answers:

1.

mkdir /etc/skel/bin

for number in $(seq 1 40); do if [ "$number" -lt 10 ]; then useradd -u 300${number} -m user${number}; else useradd -u 30${number} -m user${number}; fi; done

Best solution questions 1-3:

mkdir /etc/skel/bin

for number in $(seq -w 1 40); do

useradd -u 30$number user$number

echo "user$number" | passwd --stdin user$number

chage -d 0 user$number

done

2.

for number in $(seq 1 40); do echo user${number} | passwd --stdin user${number}; done

3.

for number in $(seq 1 40); do chage -d 0 user${number}; done

4.

at /etc/bashrc since an alias needs to be configured each time a shell is started:

alias findpdfs='find . -name "\*.pdf"'

5.

at /etc/profile:

export color=green

Can reload the shell: source /etc/profile

6.

at /etc/profile:

export PATH=$PATH:$HOME/bin

Change ownership on ~/bin of not created using /etc/skel/bin before:

for number in $(seq 1 40); do chown -R user${number}:user${number} /home/user${number}/bin; done

7.

su - user17

crontab -e

0 2 \* \* \* tar -czvf /tmp/user17\_home.tar.gz /home/user17

If want to execute every couple of minutes use \*/number at the minutes column, for example every 5 minutes: \*/5

System cron can be configured:

1. /etc/crontab
2. /etc/cron.d/mycronfile

Syntax: \* \* \* \* \* RunAs cmd

Personal cron dir:

/var/spool/cron/USERNAME

Syntax: \* \* \* \* \* cmd

8.

Correct home folder permissions for users 10-40:

for number in $(seq 10 40); do chown -R user${number}:user${number} /home/user${number}; done

su - user36

Hardlink:

ln /home/user36/shopping.txt /tmp/backup\_shopping.txt

9.

su - user12

ln -s /usr/share/doc/vim-common-7.4.160 vimdocs

10.

In /etc/bashrc:

userlist=`echo user{1..10}`

for user in $userlist; do if [ "$user" == `whoami` ]; then alias full\_access='chmod 777'; fi; done

Better option that will not run the loop every time:

userlist=`echo user{1..10}`

me=`whoami`

if echo "$userlist" | grep -w $me; then alias full\_access='chmod 777';fi

Best is to write to the personal .bashrc file for each user:

for uid in `seq -w 1 10`; do echo "alias full\_access='chmod 777'" >> /home/user$uid/.bashrc; done

11.

groupadd sales

for number in $(seq 1 10); do usermod -aG sales user${number}; done

12.

groupadd developers

for number in $(seq 11 20); do usermod -aG developers user${number}; done

13.

mkdir /home/sales

chown -R user1:sales /home/sales/

chmod 2770 /home/sales/

14.

setfacl -m u:user37:rx /home/sales/

15.

setfacl -m d:u:user37:rx /home/sales/

16.

chage -E `date -d '+365 days' +%Y/%m/%d` user14

Part 2 - Answers:

1) Create a gpt disk label on /dev/vdc

2) Create 3 partitions, each sized 2GB, on that disk

gdisk /dev/sdc

n

1

default

+2G

LVM Type: 8e00

3) Partition 1 will be an LVM physical volume. Partition 2 will also be an LVM physical volume. Partition 3 will be a regular XFS partition

mkfs.xfs /dev/vdc3

4) Format partition 3, and make sure it gets mounted on boot under /storage

At /etc/fstab:

UUID="2ef4436a-4a2a-4adf-b14c-38a86362c376" /storage xfs defaults 0 0

5) Create an LVM VG out of the two LVM partition. the VG name will be vg00

[root@clab2 ~]# pvcreate /dev/vdc1

Physical volume "/dev/vdc1" successfully created

[root@clab2 ~]# pvcreate /dev/vdc2

Physical volume "/dev/vdc2" successfully created

[root@clab2 ~]# pvs

PV VG Fmt Attr PSize PFree

/dev/vda2 centos lvm2 a-- 8.51g 40.00m

/dev/vdc1 lvm2 --- 100.00m 100.00m

/dev/vdc2 lvm2 --- 100.00m 100.00m

[root@clab2 ~]# vgcreate -s 4M vg00 /dev/vdc1 /dev/vdc2

Volume group "vg00" successfully created

[root@clab2 ~]# pvs

PV VG Fmt Attr PSize PFree

/dev/vda2 centos lvm2 a-- 8.51g 40.00m

/dev/vdc1 vg00 lvm2 a-- 96.00m 96.00m

/dev/vdc2 vg00 lvm2 a-- 96.00m 96.00m

[root@clab2 ~]# vgs

VG #PV #LV #SN Attr VSize VFree

centos 1 2 0 wz--n- 8.51g 40.00m

vg00 2 0 0 wz--n- 192.00m 192.00m

6) Create a new LV inside vg00 called "docs". Its size will be 512M, using 16M Physical Extent size

lvcreate -n docs -L 50M vg00

7) Use the target service(iscsi service) to share out the docs LV to desktop0

Configure target service:

[root@clab2 ~]# targetcli

targetcli shell version 2.1.fb41

Copyright 2011-2013 by Datera, Inc and others.

For help on commands, type 'help'.

/> cd backstores/fileio

/backstores/fileio> create target /dev/vg00/docs 50m

Block device, size parameter ignored

Note: block backstore preferred for best results

Created fileio target with size 54525952

/backstores/fileio> cd /iscsi

/iscsi> create

Created target iqn.2003-01.org.linux-iscsi.clab2.x8664:sn.85607e04061b.

Created TPG 1.

Global pref auto\_add\_default\_portal=true

Created default portal listening on all IPs (0.0.0.0), port 3260.

/iscsi> ls

o- iscsi .............................................................................................................. [Targets: 1]

o- iqn.2003-01.org.linux-iscsi.clab2.x8664:sn.85607e04061b ............................................................. [TPGs: 1]

o- tpg1 ................................................................................................. [no-gen-acls, no-auth]

o- acls ............................................................................................................ [ACLs: 0]

o- luns ............................................................................................................ [LUNs: 0]

o- portals ...................................................................................................... [Portals: 1]

o- 0.0.0.0:3260 ....................................................................................................... [OK]

/iscsi> cd iqn.2003-01.org.linux-iscsi.clab2.x8664:sn.85607e04061b/tpg1/

/iscsi/iqn.20...7e04061b/tpg1> cd luns

/iscsi/iqn.20...61b/tpg1/luns> create /backstores/fileio/target

Created LUN 0.

/iscsi/iqn.20...61b/tpg1/luns> cd ..

/iscsi/iqn.20...7e04061b/tpg1> pwd

/iscsi/iqn.2003-01.org.linux-iscsi.clab2.x8664:sn.85607e04061b/tpg1

/iscsi/iqn.20...7e04061b/tpg1> set attribute generate\_node\_acls=1

Parameter generate\_node\_acls is now '1'.

[root@clab2 ~]# firewall-cmd --add-port=3260/tcp --permanent

success

[root@clab2 ~]# firewall-cmd --add-port=3260/tcp

success

Configure client:

yum install -y iscsi-initiator-utils.x86\_64

Systemctl start iscsi

[root@clab1 ~]# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.122.54 --discover

192.168.122.54:3260,1 iqn.2003-01.org.linux-iscsi.clab2.x8664:sn.85607e04061b

[root@clab1 ~]# iscsiadm --mode node --targetname iqn.2003-01.org.linux-iscsi.clab2.x8664:sn.85607e04061b --portal 192.168.122.54 --login

Logging in to [iface: default, target: iqn.2003-01.org.linux-iscsi.clab2.x8664:sn.85607e04061b, portal: 192.168.122.54,3260] (multiple)

Login to [iface: default, target: iqn.2003-01.org.linux-iscsi.clab2.x8664:sn.85607e04061b, portal: 192.168.122.54,3260] successful.

8) Use NFS to export out /storage (from step 4) to desktop0 (use your local LAN ip address range on the export line)

9) On desktop0: use the iSCSI client tools to connect to the shared LV on server0. Once you managed to do that, create a GPT label on the LUN, create 1 partition from all its space, format the partition using ext4, and mount it under /docs (And of course, this should happen after a reboot as well)

10) On desktop0: mount the server0 exported /storage on your machine, also on /storage (and make sure it mounts across reboots)

Part 3 - Networking

1) Add two new NICS to both server0 and desktop0. All four NICS should be in a separate virtual LAN that only they share (in VMWare workstation, for example, create them on vmnet7 or something like that)

2) Configure teaming on both desktop0 and server0 from the pairs of devices

nmcli con add type team con-name TEAM0 ifname team0 config '{ "runner": { "name":"activebackup" } }'

nmcli con mod TEAM0 ipv4.addresses 192.168.122.169/24 ipv4.method manual

nmcli con add type team-slave ifname ens14 con-name TEAM0-slave0-ens14 master team0

nmcli con add type team-slave ifname ens15 con-name TEAM0-slave0-ens15 master team0

yum install -y NetworkManager-team

nmcli con up TEAM0

teamdctl TEAM0 state

Test configuration:

ip link set ens14 down

ip link set ens14 up

3) Set static ips for the links. desktop0 will be 192.168.0.1/24, and server0 will be 192.168.0.2/24.

4) Check that the teaming works by pinging across the link, and dropping the active NIC

## Linux Final Project - RHCSA

1. Not Required
2. Not Required
3. Connect to domain using LDAP:  
   setup  
   Check: Use LDAP and Use LDAP Authentication  
   Install whatever packages asked by setup and continue with the configuration including TLS certificate  
   Change directory: cd /etc/openldap/cacerts  
   Download the cert: wget -O http://URL  
   Can also use: system-config-authentication
4. Add IP in addition to DHCP:   
   Edit mode: nmcli con edit “System eth0”  
    View all options: print  
    View explanation of the required setting: desc ipv4.addresses  
    Set additional address: set ipv4.addresses 192.168.0.6/24  
   Another option using modify: nmcli con modify “System eth0” +ipv4.addresses 192.186.0.6/24  
   Another option: nmtui
5. Add hello=world to GRUB\_CMDLINE\_LINUX at /etc/default/grub  
   Update Grub files: grub2-mkconfig > /boot/grub2/grub.cfg  
   Verify: reboot and see it was added to boot parameters: cat /proc/cmdline
6. Display filesystem type: blkid  
   Check ACL support on ext: tune2fs -l /MOUNTPOINT  
   Or using: dumpe2fs /MOUNTPOINT  
   ACL is always supported on xfs
7. Add IPv6:  
   nmcli con mod "System eth0" ipv6.method manual ipv6.addr "face:b00c::6/32"  
   Apply the changes: nmcli con up “System eth0”
8. Verify Firewalld is enabled on boot: systemctl is-enabled firewalld
9. Create repo file: touch /etc/yum.repos.d/xeyes.repo  
   With the following content:   
   [xeyes]

name=Xeyes Repo

baseurl=http://classroom/blablabla

enabled=1

gpgcheck=1

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release  
List available repositories: yum list available  
Check what package provides xeyes: yum whatprovides \*/xeyes  
Install required package: yum install -y xorg-x11-apps-7.7-6.el7.x86\_64  
Install gpg key if required: rpm --import PATH2GPGKEY

1. find /usr -name "l\*" -size 3M -type f > /root/bigfiles.lst
2. groupadd coders  
   groupadd managers
3. mkdir /home/managers  
   1 option:  
   useradd -d /home/managers/james james  
   useradd -d /home/managers/satya satya  
   useradd -d /home/managers/tim tim  
   echo "james" | passwd --stdin james  
   echo "satya" | passwd --stdin satya  
   echo "tim" | passwd --stdin tim  
   Or:  
   for user in james satya tim; do useradd -d /home/managers/$user $user && echo “$user” | passwd --stdin $user; done
4. mkdir /home/coders  
   1 option:  
   useradd -d /home/coders/bruce bruce  
   useradd -d /home/coders/eric eric  
   useradd -d /home/coders/alan alan  
   echo "bruce" | passwd --stdin bruce  
   echo "eric" | passwd --stdin eric  
   echo "alan" | passwd --stdin alan  
   Or:  
   for user in bruce eric alan; do useradd -d /home/coders/$user $user && echo “$user” | passwd --stdin $user; done
5. usermod -a -G coders bruce  
   usermod -a -G coders eric  
   usermod -a -G coders alan  
   Or:  
   for user in bruce eric alan; do usermod -a -G coders $user; done
6. usermod -a -G managers james  
   usermod -a -G managers satya  
   usermod -a -G managers tim  
   Or:  
   for user in james satya tim; do usermod -a -G managers $user; done
7. for user in `cat /etc/passwd | tail -6 | cut -d: -f1`; do chage -M 30 $user; done  
   Or:  
   for user in james satya tim bruce eric alan; do chage -M 30 $user; done
8. for user in `cat /etc/passwd | tail -6 | cut -d: -f1`; do chage -E `date -d "+1 year" +%Y-%m-%d` $user; done  
   Or:  
   for user in james satya tim bruce eric alan; do chage -E `date -d "+1 year" +%Y-%m-%d` $user; done
9. mkdir /home/managers/documents  
   chown -R root:managers /home/managers/documents  
   chmod o+t /home/managers/documents  
   chmod g+s /home/managers/documents  
   chmod g+w /home/managers/documents  
   chmod o= /home/managers/documents  
   chmod o+t /home/managers/documents  
   setfacl -m u:eric:rx /home/managers/documents  
   setfacl -m u:alan:rx /home/managers/documents  
   setfacl -m u:james:rwx /home/managers/documents  
   setfacl -m d:u:james:rwx /home/managers/documents  
   setfacl -m d:u:tim:rwx /home/managers/documents
10. systemctl is-enabled sshd  
    Edit SSH configuration: vim /etc/ssh/sshd\_config  
    Add the line: PermitRootLogin no  
    systemctl restart ssh  
    Exit and Login as eric: exit  
    su - eric  
    Create SSH Key under Eric’s profile: ssh-keygen  
    Copy the SSH key to Alan’s account: ssh-copy-id alan@clab2
11. Edit cron jobs: crontab -e  
    Add the line: 0 2 \* \* \* tar cvzf /var/backups/managers\_backup.tar.gz /home/managers  
    Another option is to create a script under /etc/cron.d with the text:  
    0 2 \* \* \* root tar cvzf /var/backups/managers\_backup.tar.gz /home/managers  
    Run command every 2 seconds: watch -d COMMAND
12. Login as satya to create the .bashrc file  
    Edit .bashrc for satya: vim /home/managers/satya/.bashrc  
    Add the line: alias findstr=’egrep’
13. Login to Desktop GUI with the user tim and go to: *System tools -> Startup Applications* menu and add xeyes in the Startup Programs  
    From CLI:  
    cd to .config/autostart and add
14. mkdir /mnt/classroom  
    Edit /etc/fstab with the line:  
    classroom:/var/ftp/pub /mnt/classroom nfs defaults 0 0  
    Test mount: mount -a
15. Create PV’s: pvcreate /dev/vdc4  
    pvcreate /dev/vdc5  
    Create VG: vgcreate home /dev/vdc4 /dev/vdc5  
    Create LV: lvcreate -l 100%FREE -n homelv home  
    Format with xfs: mkfs.xfs /dev/home/homelv  
    Rename old /home dir: mv /home /homeold  
    Create new /home: mkdir /home  
    Copy contents from /homeold to /home: cp -avr /homeold/\* /home  
    Or: rsync -A -a -r /homeold/\* /home   
    Get block id: blkid  
    Edit fstab: vim /etc/fstab  
    Add the line while allowing fsck during boot and backup options: UUID="31f65f0c-98c6-4a16-8675-311e15a25624" /home xfs defaults 1 1  
    Mount it: mount -a  
    Mount it: restorecon -RFvv /home
16. Install apache: yum install -y httpd  
    Start web server: systemctl start httpd  
    Selinux: No need to enable SELinux as using default directory  
    Allow on firewall: firewall-cmd --add-service=http  
    Add permanent firewall exception: firewall-cmd --add-service=http --permanent  
    Create index.html file: vim /var/www/html/index.html  
    Test access: wget <http://localhost>
17. Install the setroubleshoot package: yum install -y setroubleshoot
18. Locate default /etc/hosts type: cat /etc/selinux/targeted/contexts/files/file\_contexts | grep "/hosts"  
    Check current SELinux on /etc/hosts: ls -Z /etc/hosts  
    Apply default selinux context: semanage fcontext -a -t net\_conf\_t “/etc/hosts[^/]\*”  
    Apply the new SELinux settings: restorecon -RFvv /etc/hosts
19. Create new partition: fdisk /dev/vdc  
    Reload new partition: partprobe  
    Format with ext4: mkfs.ext4 /dev/vdc6  
    Configure with acl by default: tune2fs -o acl /dev/vdc6  
    View mount options and verify acl enabled: tune2fs -l /dev/vdc6 | grep acl  
    Make mount point: mkdir /storage2  
    Display UUID: blkid  
    Edit /etc/fstab with the new line including the acl option:   
    UUID="25011da5-103b-44df-be2b-7c5cb3ae6a6a" /storage2 ext4 acl 0 0  
    Test mount: mount -a
20. ln -s /etc/redhat-release /etc/skel/os\_version
21. Copy service conf file to /etc/ to avoid file override during upgrades:   
    cp /usr/lib/tmpfiles.d/tmp.conf /etc/tmpfiles.d/tmp.conf  
    Edit service conf file /etc/tmpfiles.d/tmp.conf With:   
    v /tmp 1777 root root 2d  
    Restart cleanup service: systemctl restart systemd-tmpfiles-clean.service
22. Editted /etc/logrotate.conf with the following lines:

# rotate log files weekly

monthly

# keep 4 weeks worth of backlogs

rotate 3

1. Install at: yum install -y at  
   Configure to load automatically: systemctl enable atd  
   Start at: systemctl start atd  
   Create .sh file: echo "ls -la /tmp > /root/tmp\_files\_list" > atjob.sh  
   Schedule at: at -f /root/atjob.sh midnight  
   Can also schedule: at midnight  
   at> ls -la /tmp > /root/tmp\_files\_list  
   at> [ctrl+d to save]  
   Display queue: atq  
   Display at job syntax: at -c 1
2. Add static route: route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.0.254  
   Add static route to maintain across reboots: nmcli con edit “System eth0”  
   set ipv4.routes 192.168.1.0/24 192.168.0.254  
   Save  
   Reload the connection: nmcli con up “System eth0”
3. Installed autofs: yum install -y autofs  
   Display NFS exports at classroom server: showmount -e classroom  
   Create and Configure /etc/auto.master.d/ldap.autofs with:  
   /home/guests /etc/auto.ldap  
   Create and Configure /etc/auto.ldap with:  
   \* -rw classroom:/home/guests/&  
   Enable and start autofs: systemctl enable autofs  
   systemctl start autofs  
   Login as ldapuser6 and validate home directory
4. Create the errata.repo file  
   Updated the kernel: yum install -y kernel  
   Check current kernel version: uname -r  
   Locate boot menu entries: grep entry /boot/grub2/grub.cfg  
   Set the one required: grub2-set-default 'CentOS Linux 7 (Core), with Linux 3.10.0-229.14.1.el7.x86\_64'  
   Another option, less preferred:  
   Editted /etc/grub.conf so that the default boot will use the original current (in our case: menu 2)  
   Updated grub conf: grub2-mkconfig > /etc/grub2.cfg
5. Added the following line to crontab -e:  
   \*/2 \* \* \* \* ping -w 1 -c 1 192.168.0.254 || systemctl restart NetworkManager
6. systemctl set-default multi-user.target
7. systemctl disable avahi-daemon

## Linux Final Project - RHCE

PDF link: <https://www.dropbox.com/home/RHEL7/Modul-5%20%28RH199%29/HomeWorks?preview=Final_Project_RHCE7_Apr_2016.pdf>

1. Server side:  
   Configure LDAP connection and Kerberos using: system-config-authentication  
   Save keytab under /etc: wget -O /etc/krb5.keytab <http://content.example.com/pub/keytabs/server6.kytab>   
   Verify SELinux - should be keytab\_t: lz -lZ /etc/krb5.keytab  
   Create documents folder: mkdir /documents  
   Configure /etc/exports: /documents \*.example.com(sec=krb5p,ro,async)  
   Start nfs services:   
   systemctl enable nfs-server  
   systemctl start nfs-server  
   systemctl enable nfs-secure-server  
   systemctl start nfs-secure-server  
   Apply exports: exportfs -rv  
   Add to firewall: firewall-cmd --add-service=nfs  
   firewall-cmd --add-service=nfs --permanent  
   Client side:  
   Configure LDAP connection and Kerberos using: system-config-authentication  
   Save keytab under /etc: wget -O /etc/krb5.keytab [http://content.example.com/pub/keytabs/desktop6.kytab](http://content.example.com/pub/keytabs/server6.kytab)   
   Verify SELinux - should be keytab\_t: lz -lZ /etc/krb5.keytab  
   Start nfs services:   
   systemctl enable nfs-secure  
   systemctl start nfs-secure  
   Mount NFS:   
   mount -o sec=krb5p server6:/documents /mnt/serverdocs
2. Null
3. Server side:  
   Create users: useradd salsa  
   useradd tango  
   Add to Firewall (will add the newer tcp 139 and 445 ports and the backwarsd compatibility udp 137 and 138 ports): firewall-cmd --add-service=samba --permanent  
   firewall-cmd --add-service=samba  
   Install samba share: yum install -y samba samba-client  
   Make dropbox folder: mkdir /dropbox  
   Set SELinux on dropbox folder: semanage fcontext -a -t samba\_share\_t '/dropbox(/.\*)?'  
   Apply SELinux on dropbox: restorecon -vvFR /dropbox  
   Allow share home directories using samba: setsebool -P use\_samba\_home\_dirs=on  
     
   Edit /etc/samba/smb.conf:  
   workgroup = CLASSROOM  
   netbios name = SAMBA-6  
   [homes]

comment = Home Directories

browseable = no

writable = yes

valid users = salsa tango

[DROPBOX]

browseable = yes

writable = yes

path = /dropbox  
   
 Start the services: systemctl enable smb nmb  
 systemctl start smb nmb  
 Create the smb group: groupadd -r sambausers

Configure settings on the share: chgrp sambausers /dropbox

chmod 2775 /dropbox

Add to sambausers group: usermod -s /sbin/nologin -G sambausers salsa  
 usermod -s /sbin/nologin -G sambausers tango  
 Map SMB users to local users: smbpasswd -a salsa  
 smbpasswd -a tango  
 View SMB settings: testparm  
 Or: smbstatus  
 Or: smbclient -L server6  
 Clientside:  
 Install cifs utils: yum install -y samba-client cifs-utils

Make dirs: mkdir /mnt/dropboxsmb

Mkdir /mnt/salsahome  
 Test mount: mount -o username=salsa //server6/DROPOBOX /mnt/dropboxsmb  
 mount -o username=salsa //server6/salsa /mnt/salsahome  
 Can connect without mount using: smbclient

1. Install postfix: yum install -y postfix  
   Edit postfix configuration:  
   Relay server: postconf -e "relayhost=[classroom.example.com]"

Listen only on email locally: postconf -e "inet\_interfaces=loopback-only"

Allow send from server only: postconf -e "mynetworks=127.0.0.1/8 [::1]/128"

Use domain name as origin: postconf -e "myorigin=domain6.example.net"

Disable support to any incoming mail: postconf -e "mydestination="

Error for local delivery: postconf -e "local\_transport=error: local delivery disabled"

Restart postfix to apply settings: systemctl restart postfix

1. Install Apache: yum install -y httpd  
   Create directories:   
   mkdir -p /sites/abc/html  
   mkdir -p /sites/foo/html  
   Copy index.html to new directories:   
   cp /var/www/html/index.html /sites/abc/html/index.html  
   cp /var/www/html/index.html /sites/abc/html/index.html  
   Verify SELinux settings on all directories:   
   semanage fcontext -a -t httpd\_sys\_content\_t '/sites/abc/html(/.\*)?'  
   restorecon -vvFR /sites/abc/html  
   ls -Zd /sites/abc/html  
   semanage fcontext -a -t httpd\_sys\_content\_t '/sites/foo/html(/.\*)?'  
   restorecon -vvFR /sites/foo/html  
   ls -Zd /sites/foo/html   
   Add virtual sites in /etc/httpd/conf.d/site00.conf:  
   <VirtualHost \*:80>

DocumentRoot "/sites/abc/html"

ServerName s6.example.com

# Other directives here

</VirtualHost>

<Directory "/sites/abc/html">

AllowOverride None

# Allow open access:

Require all granted

</Directory>  
 /etc/httpd/conf.d/site01.conf:  
 <VirtualHost \*:80>

DocumentRoot "/sites/foo/html"

ServerName www6.example.com

# Other directives here

</VirtualHost>

<Directory "/sites/foo/html">

AllowOverride None

# Allow open access:

Require all granted

</Directory>  
 /etc/httpd/conf.d/site02.conf:  
 <VirtualHost \*:80>

DocumentRoot "/var/www/html"

ServerName server6.example.com

# Other directives here

</VirtualHost>

<Directory "/var/www/html">

AllowOverride None

# Allow open access:

Require all granted

</Directory>  
Allow http access on the firewall:  
firewall-cmd --add-service=http --permanent  
firewall-cmd --add-service=http  
Configure /etc/hosts on client and server:   
192.168.122.176 s6.example.com www6.example.com server6.example.com  
Verify SELinux boolean is on: getsebool -a | grep cgi

Verify: httpd\_enable\_cgi --> on  
Create cgi script /var/www/cgi-bin/passwd.sh:   
#!/bin/bash

echo "Content-type: text/html"

echo ''

echo `cat /etc/passwd`  
Add execute settings on the script: chmod +x /var/www/cgi-bin/passwd.sh  
Configure httpd to start automatically: systemctl enable httpd  
Start httpd: systemctl start httpd  
Test: elinks s6.example.com/cgi-bin/passwd.sh  
elinks www6.example.com/cgi-bin/passwd.sh  
elinks server6.example.com/cgi-bin/passwd.sh

1. #!/bin/bash  
   if [[ "$1" == "good morning" ]];  
   then  
    echo "Same to you"  
   elif [[ "$1" == "mud" ]];  
   then  
    echo "kip"  
   else  
    echo "Error: no parameter specified"  
    exit 1  
   fi  
   View last exit code: echo $?
2. Setup iScsi server:  
   Allow Firewall access to port 3260/tcp  
   Create new lv sized 100M: fdisk /dev/vdb  
   pvcreate /dev/vdb1  
   vgcreate iscsi\_vg /dev/vdb1  
   lvcreate -n iscsi\_disk -L 100m iscsi\_vg  
   Install targetcli: yum install -y targetcli  
   Start targetcli  
   Create target block:/backstores/block create iscsitar6 /dev/iscsi\_vg/iscsi\_disk  
   Create WWN: /iscsi create iqn.2016-05.com.example.server6:iscsitar6  
   Configure ACL for desktop:   
   /iscsi/iqn.2016-05.com.example.server6/tpg1/acls create iqn.2016-05.com.example.desktop6  
   Create LUN: /iscsi/iqn.2016-05.com.example.server6/tpg1/luns create /backstores/block/iscsitar6  
   Create Target Portal with all IP’s:   
   iqn.2016-05.com.example.server0/tpg0/portals create  
   Exit targetcli  
   Start the target service  
   iScsi client:  
   Configure initiator IQN at /etc/iscsi/initiatorname.iscsi  
   InitiatorName=iqn.2016-05.com.example.desktop6  
   Start the service iscsi   
   Discover targets on server:   
   iscsiadm --mode discoverydb --type sendtargets --portal server6 --discover  
   Login to a discovered target: iscsiadm --mode node --targetname iqn.2016-05.com.example.server6:iscsitar6 --portal server6 --login  
   Execute dmesg to see the adding of the new disk  
   Create partition on the new device and format using xfs  
   List all block devices: lsblk  
   Configure /etc/fstab:   
   UUID=b9c1fcdd-6be8-4d3e-a9de-4bc2f279c744 /iscsidir xfs defaults,\_netdev 0 0
3. Install unbound: yum install -y unbound  
   Start service and set to start automatically:   
   systemctl enable unbound  
   systemctl start unbound  
   Allow on firewall:   
   firewall-cmd --add-service=dns  
   firewall-cmd --add-service=dns --permanent  
   Edit /etc/unbound/unbound.conf  
   Interface to listen on: interface 0.0.0.  
   Allow recursive queries from local subnet: access-control: 192.168.122.0/24 allow  
   Forward requests: forward-zone:  
    name: "."  
    forward-host: 192.168.122.10  
   Allow dns queries local domain: Domain-insecure “example.com”  
   Check configuration: unbound-checkconf  
   Apply configuration: systemctl restart unbound  
   Check configuration was applied: unbound-control list\_forwards
4. Install MariaDB: yum groupinstall mariadb mariadb-client  
   Start MariaDB service: systemctl enable mariadb  
   systemctl start mariadb  
   1st time wizard: mysql\_secure\_installation  
   Set root password, agree to all defaults  
   Connect to DB: mysql -p  
   List databases: show databases;  
   Create Database named store: create database sales;  
   Change to new db: use sales;  
   Create clients table: create table clients (client\_id int(6) auto\_increment primary key, client\_name varchar(20) not null, client\_address varchar(100));  
   Create User john: create user 'john'@'localhost' identified by 'john';  
   Add permissions: grant update,insert,select,delete on sales.clients to 'john'@'localhost';  
   Display user permissions: show grants for 'john'@'localhost';  
   Update changes during runtime: flush privileges;
5. Install required network manager plugin: yum install -y NetworkManager-team  
   Update NetworkManager with the new plugin: systemctl restart NetworkManager  
   Display all cards, regardless of status: ip link  
   Add new team connection using ens and ens10:   
   nmcli con add type team con-name TEAM0 ifname team0 config '{ "runner": { "name":"activebackup" } }'  
   Configure IP: nmcli con mod TEAM0 ipv4.addresses 192.168.122.133/24 ipv4.method manual  
   Add slave NICs:  
   Add ens9: nmcli con add type team-slave ifname ens9 con-name TEAM0-slave0-ens9 master team0  
   Add ens10: nmcli con add type team-slave ifname ens10 con-name TEAM0-slave0-ens10 master team0  
   Start team: nmcli con up TEAM0  
   View status: teamdctl TEAM0 state  
   Verify teaming  
   Stop interface ens9: ip link set ens9 down  
   Start interface ens9: ip link set ens9 up  
   Stop interface ens10: ip link set ens10 down  
   Start interface ens10: ip link set ens10 up