1.Understand and use essential tools

1. **Access a shell prompt and issue commands with correct syntax**
2. **Use input-output redirection (>, >>, |, 2>, etc.)**
3. **Use grep and regular expressions to analyze text**
4. **Access remote systems using ssh**
5. **Log in and switch users in multiuser targets**
6. **Archive, compress, unpack, and uncompress files using tar, star, gzip, and bzip2**
7. **Create and edit text files**
8. **Create, delete, copy, and move files and directories**
9. **Create hard and soft links**
10. **List, set, and change standard ugo/rwx permissions**
11. **Locate, read, and use system documentation including man, info, and files in /usr/share/doc**

When you are working from a shell, an environment is created to ensure that system and user programs operate the right way. This environment consists of variables, which can be system or uyser defined. System variables have predefined names. among  them

* PATH -- set of directories in which "external" commands are searched.
* USER The name of the logged-in user
* UID The numeric user id of the logged-in user
* HOME The user's home directory
* PWD The current working directory
* SHELL The name of the shell
* PPID The process id of the process that started this process (that is, the id of the parent process)

/home/

/boot/

/etc/

/etc/sysconfig/

/lib/

/lib64/

/usr/bin/

/usr/sbin/

/usr/local/bin/

/usr/share/doc/

/opt/

/var/

/var/log/

/dev/

/sys/

/proc/

**/home/**

A user’s home directory is where the user and any application that the user runs can store information meant for exclusive use by that user. It also contains important configuration files that pertain to the user’s environment and system preferences. But because root is special, root’s home directory is /root instead of /home/root.

**/boot**

This directory contains the bootloader configuration and other files needed to boot the system, including the main kernel image and the initial ramdisk environment that bootstraps the system at startup.

**/etc**

The /etc/ directory is the main system directory for configuration files.

**/lib**

The /lib/ directory is meant to store system library files, while /usr/bin/ and /usr/sbin/ hold important userland system utilities. The /usr/local/bin/ directory is a place where users may store their own executables that they build or install from their own accounts. Otherwise, if permitted, such files may be stored in the user’s own home directory. In any case, it is good system hygeine to segregate user created tools from those on which the system depends to function.

**/usr/share/doc**

The /usr/share/doc/ directory contains useful information on installed commands and applications.

**/opt**

The /opt/ directory may be used by some applications to hold installed files, while /var/ can also hold application information but also contains a lot of other application and cache data, including system logs.

**/dev**

The /dev/ directory contains files that map to all installed and recognized devices on the system at any given time.

**/sys/ and /proc**

Filesystems /sys/ and /proc/ are created on demand, in memory, to provide information on the system as its running. In these directories, you will find bountious information regarding things like kernel runtime parameters, networking, statistics, and mounted filesystems, to name just a small few. Becoming familiar with them will eventually help you to diagnose and tune your system like a pro.

**2.Use input-output redirection (>, >>, |, 2>, etc.)**

|  |  |
| --- | --- |
| **Purpose** | **Command** |
| redirect std output to filename | > filename or 1> filename |
| append std out to filename | >> filename |
| append std out and std err to filename | >> filename 2>&1 or 1>> filename 2>&1 |
| take input from filename | < filename or 0 < filename |
| redirect std error to filename | 2> filename |
| redirect std out and std error to filename | 1> filename 2>&1 or > filename 2>&1 |

**Standard streams of information:**

stdin (0)- (Standard Input) – What you type on keyboard  
stdout (1)- (Standard Output) – Results from what you type  
stderr (2)- (Standard Error) – Any errors that occur

**Redirection symbols:**

**> Redirect output.**

Example: `echo "testing redirection of stdout" > example.txt`

\* Will input the output of echo to example.txt

\* will replace the contents of example.txt if example.txt already exists.

**>> Redirect output appending existing data.** echo "Testing redirection of stdout - Appended " >> example.txt`

Example: `

\* will be appended to the contents of example.txt.

**2> or &> – Redirect standard error**

Example: `cat ex1 ex2 1> ex3 2> ex4` contents of ex1 and ex2 are sent to ex3 and any error messages are sent to ex4.

**2>&1 – Redirect standard error to standard out**

This can be placed only at the end of the command

**< – Direct alternative input**

`wc -l < ex1`

**| – Pipes output from a command/program to the input of another command/program.**

`ls -al | grep \*.sh`

**xargs – Run a command multiple times using a list of arguments. Usage syntax:**

xargs [options] command [initial arguments]

**tee – Redirect output to a file while also allowing that output to continue to the display** **(standard output).**

`echo "tee this infor" | tee file.txt`

**chaining redirector**

`find / -iname '\*.sh' 2> /dev/null > output.txt`

Disposing error messages:

To dispose of error messages, the stderr can be redirected to /dev/null.

**3**.**RHCSA-Objective01-03: Use grep and Regular Expressions to Analyze Text**

Regular expressions alone is a lot of content. Below are the ones I find myself using the most. This is by no means comprehensive list and I would recommend studying further on grep and regular expressions due to there usefulness doing system administration.

grep

Most useful grep options/flag

-i - ignore case `grep -i '^user$'` - anything htat starts or ends with user

-r - recursive search from given path `grep -r '\*.sh' /home/greplog` - searches home directory of user greplog for files that end with .sh

-v - does not match `grep -v '^#'` - anything that does not start with a hash

-n - shows line number of match

-w - search for whole word, not words that contain the letter or combo of letters - grep -iw "is"

-A <#> - the number of lines after the greped line to display

-B <#> - the number of lines before the greped line to display

-C <#> - the number of lines before and after the greped line to display

REGEX

Regex Anchors  
\* ^ – line start with  
\* > ^e – line starts with e  
\* $ – ends with  
\* >ing$ – ends with ing

Regex Quantifiers  
\* \*  
\* +  
\* ?  
\* {}

Regex Or Operator |  
a(b|c) – mataches a string that has a followed by b or c

[] a[bc] – mataches a string that has a followed by b or c

“[0-9][0-9]:[0-9][0-9][ap]m”

Classes

\d - digit

\w - word

\s - whitespace - tabs / linebreaks

. - any character / placeholder for any character

list by file type  
ls -alr | grep “.html$”

? – matches at least one time  
+ – matchs one or more times grep -E ‘(a)+’

* Syntax: ls <option> <argument>
* Options:  
  -l : Long list including attributes  
  -a : All files and directories including hidden   
  -d : To check for a particular file or directory  
  -R : Recursively, to see the contents in a tree structure
* How to check default boot target
* How to manually switch between different targets
* How to set default boot to multi-user target
* How to set default boot to graphical target

**4. Access remote systems using ssh**

* **# ssh [-p <port>] [user@]<host|ip>**
* **# ssh <host|ip>**

# systemctl get-default

graphical.target

OR

# ls -l /etc/systemd/system/default.target

# systemctl set-default multi-user.target

# systemctl set-default multi-user

# systemctl set-default runlevel3.target

In the below example we will temporarily change from the graphical runlevel to multi-usertarget. 

1. Change to multi-user runlevel:

# systemctl isolate multi-user

# systemctl set-default runlevel3

* What are symbolic (soft) links
* What are hard links
* How to create a symbolic link
* How to create a hard link
* How to remove link

| **Target** | **Requirement** | **Octal Representation** | **Symbolic Representation** |
| --- | --- | --- | --- |
| User ( owner ) | Read, Write and Execute | 7 as in (4 + 2 + 1) | u=rwx |
| Group | Read and Write | 6 as in (4 + 2) | g=rw |
| Others | No permissions | 0 | o= |

OCTAL:

$ chmod 760 dir

SYMBOLIC:

$ chmod u=rwx,g=rw,o= dir

**ls -l**

**Locate, read, and use system documentation including man, info, and files in**

**The man pages**

To get some information about a command (here **ps**), type:

# **man ps**

Sometimes several man pages exist for a command (here **ps**), to list them, type:

# **whatis ps**

ps (1) - report a snapshot of the current processes

ps (1p) - report process status

If you are not sure about the command name and want to search for a string (here **ps**), type:

# **apropos ps**

ca**ps**h (1) - capability shell wrapper

...

getpt (3) - open the **ps**eudo-terminal master (PTM)

Each time a new package is installed, the associated man pages are added but the index database is not directly updated. This task is executed by **cron** once a day (see the **/etc/cron.daily/man-db.cron** file).

If you want to update this index database used by the **whatis** and **apropos** commands without waiting for the **cron** update, type:

# **mandb**

Note: The **makewhatis** command available in **RHEL 4/5/6** has been replaced with the **mandb** command.

**The info command**

Also, some commands (here **ipc**) offer additional documentation that you can get as follows:

# **info ipc**

To get the list of these commands, type:

# **ls /usr/share/info**

accounting.info.gz gawk.info.gz libc.info-10.gz make.info-2.gz

as.info.gz gccgo.info.gz libc.info-11.gz make.info.gz

bash.info.gz gcc.info.gz libc.info-12.gz nano.info.gz

bc.info.gz gccinstall.info.gz libc.info-13.gz nettle.info.gz

binutils.info.gz gccint.info.gz libc.info-14.gz parted.info.gz

chrony.info.gz gettext.info.gz libc.info-1.gz pinentry.info.gz

configure.info.gz gnupg.info-1.gz libc.info-2.gz pinfo.info.gz

coreutils.info.gz gnupg.info-2.gz libc.info-3.gz rluserman.info.gz

cpio.info.gz gnupg.info.gz libc.info-4.gz sed.info.gz

cpp.info.gz gprof.info.gz libc.info-5.gz standards.info.gz

cppinternals.info.gz grep.info.gz libc.info-6.gz tar.info-1.gz

dc.info.gz grub2-dev.info.gz libc.info-7.gz tar.info-2.gz

diffutils.info.gz grub2.info.gz libc.info-8.gz tar.info.gz

dir gzip.info.gz libc.info-9.gz time.info.gz

ed.info.gz history.info.gz libc.info.gz wget.info.gz

find.info.gz info.info.gz libgomp.info.gz which.info.gz

find-maint.info.gz info-stnd.info.gz libidn.info.gz

gawkinet.info.gz ld.info.gz make.info-1.gz

**The locate command**

In addition, the **locate** command provides results similar to the **find / -name** command and uses a dedicated index database called **mlocate.db**. This database is updated by **cron** once a day (see the **/etc/cron.daily/mlocate** file).

To search for a specific file (here **sysctl.conf**), type:

# **locate sysctl.conf**

/etc/sysctl.conf

/etc/sysctl.d/99-sysctl.conf

/usr/share/man/man5/sysctl.conf.5.gz

To update the **mlocate.db** index database without waiting for the **cron** update, type:

# **updatedb**

**The packages documentation**

Also, don’t forget how to find the package name providing a specific command (here **mandb**):

# **yum whatprovides \*/mandb**

...

**man-db**-2.6.3-9.el7.x86\_64 : Tools for searching and reading man pages

Repo : @anaconda

Matched from:

Filename : /usr/bin/mandb

Note: The search is not only local but also in all the remote, currently configured repositories.

And, to search for a string (here **sysctl.conf**) in a package (here **initscripts**), type:

# **rpm -ql initscripts | grep sysctl.conf**

/etc/sysctl.conf

/etc/sysctl.d/99-sysctl.conf

Note1: The use of the **locate** command should be quicker.  
Note2: Use the **-qpl** option with the complete path, if the package hasn’t still been installed.

Finally, most packages come with their own documentation in the **/usr/share/doc** directory.

Note: Most of the **SELinux** documentation is not installed by default any more, you need to install it yourself. Hopefully, you can follow the tutorial about [deploying SELinux man pages](https://www.certdepot.net/rhel7-deploy-selinux-man-pages/).

You can also get the various **man pages** and documentation available for a given package (here **less**):

# **rpm -qd less**

/usr/share/doc/less-458/LICENSE

/usr/share/man/man1/less.1.gz

/usr/share/man/man1/lessecho.1.gz

/usr/share/man/man1/lesskey.1.gz

**Additional useful commands**

Also, various commands can provide some help.

The **which** command gives you the system path associated with a command by relying on your **PATH** environment variable.

# **which mandb**

/usr/bin/mandb

The **whereis** command displays the system path of a command, the associated man page file and other files with this name known by the system. It relies on your **PATH** and **MANPATH** environment variables but not only.

# **whereis httpd**

httpd: /usr/sbin/httpd /usr/lib64/httpd /etc/httpd /usr/share/httpd /usr/shar

f you want to find the doc files for a given command (semanage for example):  
rpm -qd $(rpm -qf $(which semanage))