SysAdmin Notes for RHCSA

Somenath Sinha

August 2017

Chapter 1

Using Essential Tools

1.1 Man Command

man followed by *keyword* yields the manual page of that command.

```
$ man ls
```

man followed by option **-k** (for keyword) and then followed by a *keyword* yields a list of all the commands containing that keywork and a brief descripiton of that command.

```
[somu@testBed ~]$ man -k day
daylight (3)         - initialize time conversion information
dysize (3)           - get number of days for a given year
daylight (3p)        - set timezone conversion information
gettimeofday (2)     - get / set time
gettimeofday (3p)    - get the date and time
motd (5)             - message of the day
Net::Time (3pm)      - time and daytime network client interface
settimeofday (2)     - get / set time
Time::HiRes (3pm)    - High resolution alarm, sleep, gettimeofday,
↪  interval timers
```

The numbers next to the commands indicate which seciton of the man pages the command belongs to (based on their functionality). The actual section that the commands belong to can be determined by the use of

```
$ man man-pages
```

The relevant sections for SysAdmins are Section 1, 5 & 8. The sections are:

Section Number	Deals with	Description
1	Commands (Programs)	Those commands that can be executed by the user from within a shell.
2	System calls	Those functions which must be performed by the kernel.
3	Library calls	Most of the libc functions.
4	Special files (devices)	Files found in /dev.
5	File formats and conventions	The format for /etc/passwd and other human-readable files.
6	Games	
7	Overview, conventions, and miscellaneous	Overviews of various topics, conventions and protocols, character set standards, and miscellaneous other things.
8	System management commands	Commands like mount(8), many of which only root can execute.

To filter down the output of the **man -k** command, we can use **grep** to obtain only the relevant parts of the result on the basis of the appropriate section number in the man-pages.

This can be achieved using the pipe which feeds the output of the first command to the input of the second command.

```
[somu@testBed ~]$ man -k day | grep 3
daylight (3)          - initialize time conversion information
dysize (3)            - get number of days for a given year
daylight (3p)         - set timezone conversion information
gettimeofday (3p)     - get the date and time
Net::Time (3pm)       - time and daytime network client interface
Time::HiRes (3pm)     - High resolution alarm, sleep, gettimeofday,
↪  interval timers
```

1.2 Understanding Globbing and Wildcards

*****	-	Indicates any string.
?	-	Indicates any single character.
[· · ·]	-	Indicates any character provided within brackets.
[!· · ·]	-	Indicates any character *NOT* provided within brackets.
[a-f]	-	Indicates any character provided within the range of a to f.

1.3 Understanding Globbing and Wildcards

$ ls a*	-	Lists all files and folders (including contents of each folder) that start with "a"
$ ls *a*	-	Lists all files and folders that contain the string "a".
$ ls -d a*	-	Shows all files and folders that start with "a" but excludes the contents of each individual folder.
$ ls ??st*	-	Lists all files and folders that have "st" as the 3^{rd} and the 4^{th} character in their name.
$ [a-f]	-	Indicates any character provided within the range of a to f.

1.4 Understanding I/O Redirection and Pipes

1.4.1 I/O Redirection

File Descriptors:

STDIN	-	0	-	Standard Input	-	Represents the "file" for the Standard Input Device (generally Keyboard).
STDOUT	-	1	-	Standard Output	-	Represents the "file" for the Standard Output Device (generally the Monitor).
STDERR	-	2	-	Standard Error	-	Represents the "file" for the Standard Output Device (also, generally the Monitor).

Redirection:

STDIN	-	$<$	-	Feeds the file to the right of the "<" as input to the command on the left.
STDOUT	-	$>$	-	Stores the output of the command to the left of the ">" to the file indicated on the right. *OVERWRITES* the mentioned file.
STDOUT	-	$>>$	-	Stores the output of the command to the left of the "$>>$" to the file indicated on the right. *APPENDS* the mentioned file.
STDERR	-	$2>$	-	Redirects the errors from the command mentioned on the left to the file on the right. *OVERWRITES* the mentioned file.

```
1               $ mail -s hi root < .
2               $ ls > myFile
3               $ ls -lh >> myFile
4               $ grep hi * 2> /dev/tty6
```

1 - *mail* is a simple command used to send messages. The command expects the message to terminate with a ".", so we feed it directly to the command, instead of providing any input.

4 - The STDERR is redirected to tty6 (a virtual terminal connected to the host). Can also be diverted to a file if needed, such as an errorLog.

1.4.2 Piping

The command `$ ps aux` shows us the overview of all the running processes on the host. However, it's too long to view all at once. In such situations, or wherever we need to feed the output of the first command to the input of the second command, we use the pipe operator. The command would then be `$ ps aux | less`.

The difference in the usage of the piping and redirection operators is that Pipe is used to pass output to another program or utility, while Redirect is used to pass output to either a file or stream.

1.5 Using I/O Redireciton and Piping

```
$ ps aux | awk '{print $2}'
$ ps aux | awk '{print $2}' | sort
$ ps aux | awk '{print $2}' | sort -n
```

The second column ($2) of the `$ ps aux` command contains the Process ID (PID), and if we only want to filter the output such that only the PID is shown, we simply use the **awk** filtering utility.

If we want to sort the output of the command, we use the **sort** utility, but it generally sorts as a string. To sort the output as a number, we use the option **sort -n**.

If you expect lots of errors for a particular command, but want to discard all errors and only see the output when successful, then simply redirect the STDERR to `/dev/null`, which is a special device that discards all data written to it, i.e., a dustbin for data.

```
$ find / -name "*.rpm"
$ find / -name "*.rpm" 2> /dev/null
```

The first command shows all output including errors, but the second command discards all errors and shows the rest.

```
$ some_command > /dev/null 2> &1
```

The above code redirects STDOUT to `/dev/null` thus destroying the output, and also redirects the STDERR (2>) to STDOUT (&1). Essentially, it discards all output - useful when we don't need the output but only need the command to execute.

```
$ ls / > file_list.txt
$ sort < file_list.txt > file_list_sorted.txt
```

The above command stores the contents of the root directory in *file_list.txt*. Then, the second command uses both input and output redirection! The input of the sort command is fed from *file_list.txt* and the corresponding output sent to *file_list_sorted.txt*.

Chapter 2

Essential File Management Tools

2.1 Understanding Linux File System Layout

root (/)	-	Contains all other directories.
/boot	-	Contains everything the system needs to start up
/usr	-	Contains program files
/etc	-	Contains configuration files
/home	-	Contains a user's files
/mnt	-	Used to manually mount devices
/media	-	Devices like optical discs get auto-mounted on the media directory

Unlike other OSs, the linux files system is designed as such that multiple devices can be mounted on the same file system hierarchy. Thus, it's possible to mount devices remotely as well!

2.2 Finding Files

The **find** command is used to find a file within a folder and its subdirectories. When the starting point of the search is the root directory (/) then find will search the entire file system. While the utility is extremely thorough, this may cause delays due to remote devices on the network mounted on the file system.

```
$ find / -name "passwd"
```

If you're trying to find the location of a binary file, a better command would be **which** command, as it directly shows the location of the binary, but be careful as it only works with binaries.

```
$ which passwd
/usr/bin/passwd
```

Contrastingly, the command **whereis** not only gives us the locaiton of the binary, but the location of the complete environment of the binary!

```
$ whereis passwd
passwd: /usr/bin/passwd /etc/passwd /usr/share/man/man1/passwd.1.gz
↪  /usr/share/man/man5/passwd.5.gz
```

Another similar utility is called **locate** which shows all files that have the string provided to it in its name. Note, however, that locate operates on a database, that must be updated (especially after the creation of a new file) to show relevant results.

```
[root@testBed Documents]# touch sinha
[root@testBed Documents]# ls
sinha
[root@testBed Documents]# locate sinha
/usr/share/vim/vim74/keymap/sinhala-phonetic_utf-8.vim
/usr/share/vim/vim74/keymap/sinhala.vim
[root@testBed Documents]# updatedb
[root@testBed Documents]# locate sinha
/home/somu/Documents/sinha
/usr/share/vim/vim74/keymap/sinhala-phonetic_utf-8.vim
/usr/share/vim/vim74/keymap/sinhala.vim
```

2.3 Understanding Links

inode - An inode is a datastructure that describes a file system object such as a file or a directory, containing both the disc block locations as well as the attributes of the file system object. The inodes are identified by their inode number.

Consequently, for us to access the files/directories, we need to be able to provide a name to the inodes, which are called hardlinks. A file may have more than one hardlink. Note that each hardlink is simply a different name provided tot he same inode. Ths, all hardlinks to the same file/directory have the same inode number. Hardlinks are one-directional only, i.e., the hardlink itself knows which inode it points to, but the inodes only know the total number of hardlinks that are associated with it, and not which exact ones are pointing to it. Since hardlinks point to some inode, they always need to stay on the same partition as the inode.

A symbolic link on the other hand, points to a hardlink instead of an inode. As such, it has a different inode number than the one that the hardlink points to. Thus, the hardlink and symbolic link can be on different partitions as well. It can even exist across servers. Whenever a hardlink is deleted, however, all the symbolic links pointing to it are rendered invalid.

2.4 Working with Links

The `ln` command is used to create both hardlinks and symbolic links. To create a symbolic link, we need only add the `-s` option. The `-i` option of the `ls` command shows us the inode number.

```
[root@testBed somu]# ln /etc/hosts computers
[root@testBed somu]# ls -il /etc/hosts computers
```

```
8388733 -rw-r--r--. 2 root root 158 Jun  7  2013 computers
8388733 -rw-r--r--. 2 root root 158 Jun  7  2013 /etc/hosts
[root@testBed somu]# ln -s computers newcomputers
[root@testBed somu]# ls -il /etc/hosts computers newcomputers
8388733 -rw-r--r--. 2 root root 158 Jun  7  2013 computers
8388733 -rw-r--r--. 2 root root 158 Jun  7  2013 /etc/hosts
27604468 lrwxrwxrwx. 1 root root   9 Sep  7 19:26 newcomputers ->
↪  computers
[root@testBed somu]# rm -f computers
[root@testBed somu]# ls -il /etc/hosts newcomputers
8388733 -rw-r--r--. 1 root root 158 Jun  7  2013 /etc/hosts
27604468 lrwxrwxrwx. 1 root root   9 Sep  7 19:26 newcomputers ->
↪  computers
[root@testBed somu]# exit
exit
[somu@testBed ~]$ ln /etc/shadow mydata
ln: failed to create hard link ‘mydata’ => ‘/etc/shadow’: Operation not
↪  permitted
[somu@testBed ~]$ ls -l /etc/shadow
----------. 1 root root 1375 Sep  5 21:04 /etc/shadow
```

When the hardlink *computers* to the inode associated with `/etc/hosts` is deleted, the associated symbolic link of *newcomputers* becomes invalid.

Finally, RHEL 7 onwards, a user may only create a link to a file/directory that it at least has a read permission to. Thus, any user won't be able to create a link to `/etc/shadow` as it has no permissions for anybody.

2.5 Working with tar

tar stands for Tape Archive. The command is most commonly used to make backups of files by storing them in archives. Some of the options of `tar` are:

-c	-	create	-	typically has an extension of .tar
-t	-	show contents	-	show contents of the archive.
-x	-	extract		
-z	-	file	-	compress the archive using gzip. Typically has an extension of `.tgz`
-v	-	verbose	-	tell us what the utility is doing.
-f	-	file	-	option to indicate the name of the archive file.
-C	-	location	-	indicates where the archive is to be extracted.

```
$ tar -cvf /root/etc.tar /etc
```

The above command creates the `etc.tar` archive in the `/root` directory and puts the contents of `/etc` in that archive. Note that the file `etc.tar` has a `.tar` extension only because we provided it, and not because Linux mandates it (unlike windows). Thus, sometimes we may run across tar archives that don't have an extension and are hard to detect. So, in that case we use the file command, which tells us the type of a particular file.

```
$ file /root/etc.tar
/root/etc.tar: POSIX tar archive (GNU)
```

Note that the .tar archive only puts all the files of teh `/etc` directory in the file `tar.etc`, but doesn't actually compress anything. To enable compression the `-z` option of the `tar` command must be used.

```
$ tar -czf /root/etc2.tgz /etc
```

Before extracting the contents of a tar file, we might want to see its contents, which can be done using the `-t` option of the `tar` command. *NOTE: Some older versions of* `tar` *may require the -z option to enable working with gzip archives, even when simply using the archive and not creating it.*

```
$ tar -tvf /root/etc2.tgz
```

To actually extract the archive, we use `-x` option. To indicate the location where we want the extracted files to reside, we include the `-C` option. If this option is not present then the files will be extracted in the present directory.

```
$ tar -xvf /root/etc2.tgz -C /tmp
```

To extract only one file from the archive, we can simply provide the name of the file at the very end.

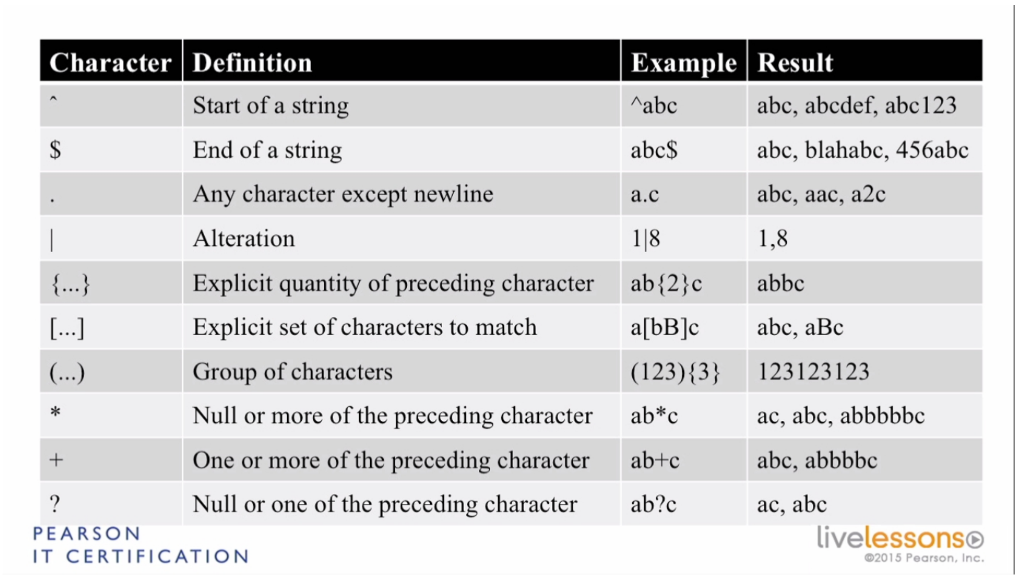
```
$ tar -xvf /root/etc2.tgz -C / etc/wgetrc
```

NOTE that in the above command, we use the relative path `etc/wgetrc` *because of the fact that the archive stores a relative file path for easy extraction in any folder.*

Chapter 3

Working with Text Files

3.1 Understanding Regular Expressions



Character	Definition	Example	Result
ˆ	Start of a string	^abc	abc, abcdef, abc123
$	End of a string	abc$	abc, blahabc, 456abc
.	Any character except newline	a.c	abc, aac, a2c
|	Alteration	1|8	1,8
{...}	Explicit quantity of preceding character	ab{2}c	abbc
[...]	Explicit set of characters to match	a[bB]c	abc, aBc
(...)	Group of characters	(123){3}	123123123
*	Null or more of the preceding character	ab*c	ac, abc, abbbbbc
+	One or more of the preceding character	ab+c	abc, abbbbc
?	Null or one of the preceding character	ab?c	ac, abc

PEARSON
IT CERTIFICATION
livelessons
©2015 Pearson, Inc.

Figure 3.1: RegEx Cheat Sheet

3.2 Using common text tools

3.2.1 cat

The `cat` command prints the entire content of a file on to the terminal.

3.2.2 less

Sometimes the `cat` command is unsuitable, like in the case of extremely large files. In such cases, like the `/var/log/messages`, the default system log file, using `cat` won't work as the majority of the messages would scroll past fast. For such cases, `less` is a better utility. Search functionality is exactly the same as in the case of vim.

3.2.3 Head and Tail

Head

The `head` command by default shows us the first 10 lines of a text file. To see more or less lines, the `-n` option can be used.

```
$ head -n 20 file.txt
```

Tail

The `tail` command by default shows us the last 10 lines of a text file. To see more or less lines, the `-n` option can be used.

```
$ tail -n 5 file.txt
```

Combination of head and tail

The combination of these two commands can enable the viewing of text in between specific line numbers. The command below shows lines 16-20 of `file.txt`

```
$ head -n 20 file.txt | tail -n 5
```

3.2.4 cut

With the `cut` utility, we can print out a specific column form a text file. It assumes the columns are separated by Tabs. Which specific column is to be printed is set by using the `-f` option. For example, to only print the first column of a text file, we say:

```
$ cut -f 1 cities
```

To provide a different delimter, such as ":" we use the `-d` option followed by the delimiter of our choice.

```
$ cut -f 1 -d : /etc/passwd
```

3.2.5 sort

This command sorts the input provided in the order of the ASCII table. That means numbers first, captial letters next and finally the lower case letters.

```
$ cut -f 1 -d : /etc/passwd | sort
```

To sort on the basis of a specific criteria:

-n	-	Sort on the basis of actual numberical value, instead of treating a number as a string.
-f	-	Sort in a case insensitive manner.

3.2.6 tr

The `tr` command replaces certain characters with certain other characters. Thus, it's frequently used in conjunction with pipes to modify the output of a command.

```
$ echo hello | tr a-z A-Z
HELLO
$ echo hello | tr [:lower:] [:upper:]
HELLO
```

3.3 grep

`grep` is a filtering utility that only prints those lines that contain a certain expression matching the pattern provided by a *RegEx*.

```
$ ps aux | grep tracker
somu      10450  0.0  0.4 469796  9000 ?       SNl  10:06   0:00
↪  /usr/libexec/tracker-miner-user-guides
somu      10465  0.0  0.6 536856 12012 ?       Sl   10:06   0:00
↪  /usr/libexec/tracker-store
somu      10611  0.0  0.7 779816 13108 ?       SNl  10:06   0:00
↪  /usr/libexec/tracker-extract
somu      10614  0.0  0.5 469800  9632 ?       SNl  10:06   0:00
↪  /usr/libexec/tracker-miner-apps
somu      10615  0.0  0.7 710160 13204 ?       SNl  10:06   0:00
↪  /usr/libexec/tracker-miner-fs
root      17396  0.0  0.0 112644   968 pts/0   R+   13:49   0:00 grep
↪  --color=auto tracker
```

Another use for `grep` is searching files. To avoid errors notifying "is a directory", simply redirect errors to `/dev/null`.

```
$ grep lisa * 2> /dev/null
group:lisa:x:1001:
gshadow:lisa:!::
passwd:lisa:x:1001:1001::/home/lisa:/bin/bash
passwd-:lisa:x:1001:1001::/home/lisa:/bin/bash
services:na-localise     5062/tcp                # Localisation access
```

```
services:na-localise     5062/udp                # Localisation access
shadow:lisa:$6$0l/zSJkh$xjJNYNnj1rPs7Fq0hDWt8VucS0nLL82XrMYpmBnLF2DrzB2npFvCwxM9MJEHgCHCwvabCgEA17L
shadow-:lisa:password:17414:0:99999:7:::
```

3.3.1 wc

Counts the number of words, lines and characters.

-l	-	Counts the number of lines
-w	-	Counts the number of words
-m	-	Counts the number of characters
-c	-	Counts the number of bytes
-L	-	Counts the length of the longest line

Chapter 4

Keyboard Shortcuts

CTRL+U	-	Clear Command Line.
CTRL+L	-	Clear Screen, same as clear command.