Linux Sys-Admin 101
Cheat Sheets

for

By Thomas Weeks, V2013-11-05

Red Hat Linux Runlevels

Starting/Stopping Services

```
# service [service-name [start | stop | restart] ]
```

OR

```
# /etc/rc.d/init.d/service-name [start | stop | restart]
```

Changing Service/Runlevel Configurations

```
# chkconfig --level [12345] service [on | off]
```

Runlevel Name	#	Runlevel Description
Halt	0	Used to stop the system without reboot. Shuts down all services.
Single User	1	No apps or services running, net-working off. Used for sys- maint.
Multi User	2	Networking on, but major net services are all off (web email...)
full m-ultiuser	3	Same as 2, but with major net-working apps running. Servers.
Unused	4	Reserved for admin's own use: customized run level, backups...
X11	5	Same as 3, but with X11 GUI running. Desktops.
Reboot	6	Level only used for rebooting the system. Shuts down all services.

Common vi/vim Commands

<esc>	return to command mode	:%s/x/y[/g]	search for x, replace w/y, g=globally
<esc>:	get into colon mode	.	repeat last command
i	insert mode	[n]x	delete character[s] on & to the right
a	append (insert) at current position	[n]dd	delete lines[s] on and below
A	append (insert) at eol	<ctrl>w,s	Split screen mode
h & l	h-move left & l-move right	<ctrl>w,_	Toggle to bottom screen
j & k	j-move down, k-move up	<ctrl>w,	Toggle to upper screen
:w	write file	<shft>v	Normal visual edit mode
:w file2	write to filename file2	<ctrl>v	Columnar visual edit mode
:w!	write to read only file	[n]y	yank (copy into buffer)
:wq	write file and quit (also :x)	p	put (from buffer) after
:q!	quit and drop changes	P	put (from buffer) before (insert)
<shift>zz	quick write + quit (same as :wq)	[n]u	undo n-many
:e file1	load and edit file1	[n]<ctrl>r	redo n-many
:/string	search for string	g / G	top / bottom of file
n	search for next	:!<comm>	execute command

Other CLI/Bash Specific Keyboard Tricks:

- Full editable command line history: Use **<up><down><left><right>** arrows **<del><bs>** for deleting
- Scrollable shell buffer: **<shift><up>/<down>/<pg-up>/<pg-down>**
- Mouse text copy/paste: Highlight test with mouse/left-button paste with **<middle>** or **<right><left>** (simultaneous) mouse button(s)
- Move beginning of line: **<ctrl><a>**
- Move to end of line:**<ctrl><e>**
- Move forward word:**<esc><f>**
- Move backward word: **<esc><b>**
- Clear Screen: **<ctrl><l>**
- Clear/cut to the left: **<ctrl><u>**
- Clear/cut to the right: **<ctrl><k>**
- Paste/insert: **<ctrl><y>**
- Reverse history search: **<ctrl><r>**
- Reverse history execute: **!<partial command>**

Important Server System Directories

/ <----The Root or top level directory, or "root filesystem"

/bin/ <----Main directory for stock system binaries. Needed for booting. ~ 4-5MiB

/boot/ <----Where boot files and kernel reside. ~10-100MiB (usually a mount point for first part.)

/dev/ <----System directory where "devices" reside (uses udev dynamic device files)

/etc/ <----Main system and app. configuration files reside. ~10-20MB

- **X11/** Where all X-Window files are stored
- **httpd/conf/** Where Apache web server config files are stored
- **init.d/** Hard link shared with /etc/rc.d/init.d/ (in RHEL)
- **mail/** Where Sendmail keeps most of its config files (sendmail.mc/cf, virtusertable, etc)
- **portsentry/** Where the portsentry config files are located (RS)
- **postfix/** Where Postfix keeps most of its config files (main.cf, master.cf, virtual, etc)
- **xinetd.d/** Where xinetd Internet sub-daemon startup files are located (/etc/xinetd.conf)
- **rc.d/** Where all run level scripts and subdirs are located (rc, rc.local, rc.sysinit)
  - **init.d/** Where the actual service scripts are located
  - **rc0.d/** Where Start/Kill symlinks are stored for halt level (links to ../init.d/<file>)
  - **rc1.d/** Where Start/Kill symlinks are stored for single user mode (RH specific levels)
  - **rc2.d/** Where Start/Kill symlinks are stored for multi user+net mode
  - **rc3.d/** Where Start/Kill symlinks are stored for multi+net+services
  - **rc4.d/** Where Start/Kill symlinks are stored for reserved mode
  - **rc5.d/** Where Start/Kill symlinks are stored for multi+net+services+X11
  - **rc6.d/** Where Start/Kill symlinks are stored for Reboot level
- **sysconfig/** Where config files and the "network" files are (RHEL)
  - **network-scripts/** Where ifcfg files for network system IPs are stored. (RHEL)
- **webmin/** Where RS puts all webmin WebGUI admin config files (RS)

/home/ <-----Where all user home directories reside from ~1MiB and up

- **httpd** Where RS symlinks to the apache/httpd doc-root /var/httpd/ (RS).

/lib/ <-----Where most system libraries and DLLs reside Space: ~100-150MiB

- **modules/2.xx.yy/** Where kernel modules are stored for kernel 2.xx.yy

/media/ <-----The new removable media mount point for cds, DVDs, USB/firewire drives, etc.

/mnt/ <-----Empty dir where other drives and file systems get mounted (e.g. includes floppy, cdrom, backup, & NFS/net-mounts)

/opt/ <-----Where optional (non-distro usually) static built software packages get installed to.

/proc/ <-----A virt-filesystem of directories for seeing & talking to the kernel.

/root/ <-----The root user's home directory (with permissions 700 or 750).

/sbin/ <-----Where system binaries and Sys-Admin programs reside (like fsck, ifup, modprobe, reboot/halt, etc. needed for booting), ~15MiB

/selinux/ <----The pseudoFS where parts of the selinux kernel system reside. (non-RS)

/srv/ <-----Where server content (FTP/WWW/CVS) will (or should) eventually reside.

/sys/ <-----The new 2.6 virt-fs that give a hierarchy view of specific hardware devices.

/tmp/ <-----Where any system/user temporary files can go. At RS fs w noexec, nosuid, nodev

/usr/ <-----Common file space. ~500MiB-2GiB

- **bin/** Common additional system & user binaries ~50-60MiB
- **lib/** Additional shared libraries ~400-500MiB
- **local/** Where other additional distro packages install to ~50-100MiB
- **sbin/** Sys-Admin secured binaries ~15MiB
- **src/** Where system source code is installed ~0-100MiB

/var/ <-----Where the system stores data, logs, variables, etc ~10MiB-xGiB

- **lib/** Where MySQL and postgressSQL keep their DB and config files
- **log/** Where most of system and apps stores log files
  - **portsentry/** Where portsentry logs to (RS)
- **named/** Where named/BIND DNS stores zone files and/or runs chrooted (RH)
- **run/** Where system stores important running program ID files
- **spool/** Where print, mail, cron, mail and other queue files are stored.
- **tmp/** Temp files that are preserved between reboots (RS maps to /tmp w/noexec)

Few of Most Common Linux/POSIX Commands

Change directory:
cd [dir]

Make directory:
mkdir [dir]

Remove directory:
rmdir dir

Copy files to path:
cp [opt] source dest

Move/rename files:
mv [opt] source dest

Print Working Dir:
pwd

Search files for string:
grep [opt] pattern [file]

Locate known files:
locate file

Find file by expression:
find [path] [expression] [options]

Edit file:
vim [file]

List/combine a file:
cat [opt] [file1]...

List last lines of a text file:
tail [opt] [file]

Read a file:
less [file]

Sort a file:
sort [file]

Reformat/cut text:
cut [opt] [file]

Print txt part of a binary:
strings [file]

Making a symlink:
ln -s target linkname

Making a hardlink:
ln target hardlinkname

Change file permissions:
chmod [perms] file...

Change owner/group:
chown owner.group file...

Command Line I/O and Manipulation

Ouput Redirection: `$ ls -la >file-out.txt`

Output Concatenate: `$ ls -la >>file-append.txt`

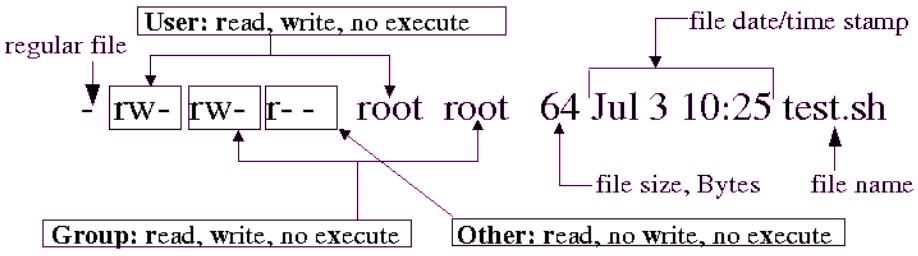
Output Piping:
```
$ cat /etc/passwd | grep bash | mail\
 -s"Shell users $HOSTNAME" admin@mydomain.com
```

Command Stacking: `$ ls -la ; date ; program3 ; program n.`

Input from File: `$ program < file-input.txt`

Background Process: `$ program &`

If 1 ran ok, run 2: `$ 1 && 2`



Examining File Types and inode data

Determine file type: `# file <filename>`

Determine file block/inode data: `# stat <filename>`

Other Common Admin Commands

Searching for process: `# ps auxw | grep <proc name>`

Searching for net port/srvc: `# netstat -antp | grep :<port>`

Run a remote command: `# ssh user@host 'command'`

Wipe a hard drive: `# dd if=/dev/zero of=/dev/sdc`

Back up MBR/part-table: `# dd if=/dev/sda of=mbr.img bs=512 count=1`

Mouting filesystems: `# mount [-o opts] <part-dev> <m.point>`

Force umounting: `# fuser -mk <m.point> && umount <m.point>`

Common File Extensions and Usage

File Extension	File Type and Description
.sh	shell script file extension. Used for third party as well as well as most home grown shell scripts. A ".sh" file in UN*X usually assumes underlying shell is "sh" (bourne), but in Linux it is usually bash. See m*an bash*
.shar/.tar.sh	A shell script (usually install script) with an embedded uuencoded tar ball file. You can extract the UU encoded section, decode it, and then untar it (usually). But it is mean to be run from the shell. Will usually start with a standard script line of #!/bin/bash or /bin/sh.
.zip	DOS/Windows common compression program. Not always UN*X friendly, but Linux contains compatible zip/unzip programs. See *man unzip* or *zip*
.gz/.gzip	GNU-zip or gzip. Like windows zip but is better suited for UN*X/Linux, is non-licensed, fully transportable, and is installed on most modern UN*Xs. Also works works well with std-io. To create, cat file1 ... | gzip > file.gz. To decompress can use gunzip file.gz. Use tar-gz to maintain directory/permission structure. See *man gzip*
.tar	Tape Archive formatted file. Like a zip file, but has less compression, is transportable to most all UN*X OS' and can retain most common UN*X file permission attributes. Tar can create tar files, or can stream files to devices. Typically created with *tar cvf archive.tar [files/*]* and un-tar'd with *tar xvf archive.tar [files/*]*. See *man tar*
.tgz/.tar.gz	This is a tar file that has been further compressed with gzip. Typically created with *tar czvf archive.tgz [files/*].* De-compressed with *tar xzvf arcive.tgz [files]*. See *man tar*
.UU	uuencoded file. Is basically just a binary, or other file that was sent by email or usenet. The UU encoding translates the 8bit binary code down into ASCII 7bit for transmission over 7bit transport (like email or usenet). To decode, use *uudecode filename.uu*, to create, and use *uuencode filename filename >filename.uu*. See *man uuencode* and *uudecode*.
.Z	Compressed file using *compress*. Older, but still used form of compression. Use *compress file.Z* to create, *decompress file.Z* to decompress. See *man compress* for more info.
README/txt	Just a text file, usually from a programmer or developer. Can be ready by using either *more README* or *less README* or opened in favorite editor.

Most Common RPM Commands

install a package:
rpm -i <package>.rpm
Use an http or ftp URL to download and install.

test install:
rpm -<command> --test <package>[.rpm]
Test to see if an install would work (w/dependencies)

uninstall package:
rpm -e <installed packagename>

upgrade/nodeps:
rpm -Uvh <package>.rpm

query-info:
rpm -qpi <package>.rpm
-i for info, -p tells all about an uninstalled package file.

query file:
rpm -qf /path/misc-installed-file
Handy to see what installed package a file belongs to.

query-filelist:
rpm -qpl <package>.rpm
Handy to see what files are going to be installed and where.

Most Common yum Commands

install package(s):
yum [-y] install [package1] [package2] ...
the -y to auto accept the y/N prompts.

remove package(s):
yum [-y] remove [package1] [package2] ...
to remove one or more packages.

search pacakge(s):
yum search string1 [string2] ...
regular text against package name, desc. or RPM info.

update a package:
yum update [package]
no package updates everything, use `upgrade` to obsolete old packages (better for doing major rev system upgrades)

check avail. update:
yum check-update
check to see if any updates are available for your install.

whatprovides?
yum whatprovides file/command/feature1 [feature2] ...
ask what package provides a specific program.

Drives, Partitions, LVM and Filesystems

List all partitions:	`# fdisk -l`
Modify & rescan partitions:	`# fdisk <dev> && partprobe -s`
Create & display LVM pv:	`# pvcreate <dev> && pvdisplay`
Create & display LVM vg:	`# vgcreate vgname <PV-dev> && vgdisplay`
Create & display LVM lv:	`# lvcreate -l <#PEs> <vgname> <PV-dev>`
Grow a LVM vg:	`# vgextend <vgname> <PV-dev>`
Grow a LVM lv:	`# lvextend -l +<#PEs> <LV-dev> <PV-dev>`
Format filesystem ext3:	`# mkfs.ext3 <dev>`
Grow an ext3 filesystem:	`# resize2fs <dev> [newsize]`

Other Common Tools

Local Drive to Remote|Local Drive backup using rsync: (uses SSH xport & auth)

```
rsync -av --exclude=/proc/ --exclude=/dev/ --exclude=/tmp/  --exclude=/sys/ \
--exclude=/mnt/  --exclude=/media/  <source-root-dir> <remote@host|local-dest>
```

Recursive File Transfer from local to remote machine:

```
scp -p <source path or ./> user@host:/<target-path>
```