Linux Reference

Table of Contents

Introduction

A collection of Red Hat Enterprise Linux command references.

Useful adhoc commands

```
# dd if=/dev/urandom of=/stratisvol/file2 bs=1M count=2048
```

```
# dd if=image.iso of=/dev/sda status=progress
```

Remove Password from protected PDF

```
# dnf install qpdf
```

```
# qpdf --password=changeme --decrypt secure.pdf unsecure.pdf
```

Generate a UUID

```
# uuidgen
```

Find files modified in the past X days

```
# find /dir -mtime -X
```

Remove comments and blank lines from a file

```
# grep -v -e '^#' -e '^$' /etc/sysconfig/sshd > /var/tmp/sshd
```

Find lines starting with string

```
# grep -R ^root /etc/ 2> /dev/null > /var/tmp/results.txt
```

Run in background and redirect the output to file

```
# nohup command &> output.out &
```

Wait for all `systemd-udevd` events to finish

```
# udevadm settle
```

Format a USB pen drive

Delete any old partitions on the USB key:

```
Open a terminal and type sudo su
Type fdisk -l and note your USB drive letter.
Type fdisk /dev/sdx (replacing x with your drive letter)
Type d to proceed to delete a partition
Type 1 to select the 1st partition and press enter
Type d to proceed to delete another partition (fdisk should automatically select the
second partition)
Type w to write changes to the USB key
```

Create a new partition:

```
Type n to make a new partition
Type p to make this partition primary and press enter
Type 1 to make this the first partition and then press enter
Press enter to accept the default first cylinder
Press enter again to accept the default last cylinder
Type w to write the new partition information to the USB key
```

Ensure `unmount` (which it should be already):

```
# umount /dev/sdx (replacing x with your drive letter)
```

Create a fat files system (replacing x with your USB key drive letter):

```
# mkfs.vfat -F 32 /dev/sdx1
```

Label it:

```
# dosfslabel /dev/sdx1 "USB"
```

Date and Time

The Network Time Protocol (NTP) is a standard way for machines to provide and obtain correct time information on the Internet.

Use the `timedatectl` command to show an overview of the current time-related system settings.

Use the `timedatectl list-timezones` to list the available time zones.

The following `timedatectl` command updates the current time zone:

```
# timedatectl set-timezone Europe/London
```

The `timedatectl set-ntp` command enables or disables NTP synchronization.

The `chronyd` service keeps the usually-inaccurate local hardware clock (RTC) on track by synchronizing it to the configured NTP servers.

The `driftfile` specified in the `/etc/chrony.conf` configuration file records RTC clock drift if network connectivity is lost.

```
# dnf install chrony
# systemctl start chronyd
```

Add servers or pool of servers to `/etc/chrony.conf` for chrony as a source for chrony to synchronize time.

Use the `chronyc sources -v` command to verify NTP servers.

Summary

The `chronyd` service helps to synchronize time settings with a time source. The time zone of the server can be updated based on its location.

Command References:

`timedatectl`, `chronyd`, `chrony.conf`, `chronyc`

Cron

A best practice is to run recurring jobs from system accounts using system-wide `crontab` files.

The `/etc/crontab`:

```
# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

Always create custom crontab files under the `/etc/cron.d` directory.

A command called `run-parts` called from the `/etc/cron.d/0hourly` file runs the `/etc/cron.hourly/* scripts`. The run-parts command also runs the daily, weekly, and monthly jobs, called from a different configuration file called `/etc/anacrontab`. The syntax of `/etc/anacrontab` is different from the regular crontab configuration files. It includes `Period in days`,`Delay in minutes`, `Job identifier` and `Command`.

A new scheduling function is now available, `systemd timer units`.

Examples can be viewed here:

```
/usr/lib/systemd/system/*.timer
```

Never modify any unit configuration file under the `/usr/lib/systemd/system` directory directly, instead make a copy indented for change under `/etc/systemd/system`.

Modifying any `systemd` unit file required a `daemon-reload`.

```
# systemctl daemon-reload
```

Summary

Recurring system jobs execute tasks on a repeating schedule. Recurring system jobs accomplish administrative tasks on a repeating schedule that have system-wide impact.

Command References:

`crontab`, `anacrontab` and `systemd.time`.

Software Packages

`dnf` or `yum` is designed for managing RPM-based software installation and updates. The `dnf` command allows you to install, update, remove, and get information about software packages and their dependencies.

Use list and search.

```
# dnf list 'tree'
# dnf search all 'web server'
```

Detailed information about a package can be obtained with the `info` option.

```
# info httpd
```

Use the `provides` option to establish what package provides a familiar tool or program.

```
# dnf provides */semanage
```

Install packages using the `install` option and with the `-y` to bypass any prompts.

```
# dnf install policycoreutils-python-utils -y
```

Package groups can be listed and installed too.

```
# dnf group list
# dnf group install "Development Tools"
```

Using `dnf history` packages can be removed or undone.

```
# dnf history
# dnf history undo 8
```

To see all available repositories use the `dnf repolist all` command. Found in the `yum-utils` package, `yum-config-manager` can be used to enable or disable repositories.

```
# dnf install yum-utils
# yum-config-manager --enable ansible-2.8-for-rhel-8-x86_64-rpms
```

Non-Red Hat sources provide software through third-party repositories. Enable support for a new third-party repository, create a file in the `/etc/yum.repos.d/` directory. Repository configuration files must end with a `.repo` extension.

```
[EPEL]
name=EPEL 8
baseurl=http://dl.fedoraproject.org/pub/epel/8/x86_64/
enabled=1
gpgcheck=0
```

Summary

Use the `rpm` command to query a local database to provide information about the contents of installed packages.

Use the `dnf` or `yum` powerful command-line tools to install, update, remove, and query software packages.

Command References:

`dnf`, `yum`, `yum-config-manager` and `yum.conf`.

Firewall

`netfilter` is a framework for network traffic operations such as packet filtering, network address translation and port translation. The Linux kernel also includes `nftables`, a new filter and packet classification subsystem.

Firewalld is a dynamic firewall manager, a front end to the `nftables` framework.

Firewalld has pre-defined zones, each of which you can customize. Default configuration zones include `trusted`, `home`, `internal`, `work`, `public`, `external`, `dmz`, `block` and `drop`.

The `firewalld` service is controlled by `systemd`:

```
# systemctl stop firewalld
# systemctl start firewalld
# systemctl status firewalld
```

Firewalld has several pre-defined services, including the essential being `ssh`.

Use the `firewall-cmd` command-line tool to configure firewall rules.

Examples:

```
# firewall-cmd --get-default-zone
# firewall-cmd --set-default-zone=public

# firewall-cmd --permanent --add-service=ftp --zone=public
# firewall-cmd --permanent --add-service=http --zone=public
# firewall-cmd --permanent --add-service=ntp --zone=public
# firewall-cmd --permanent --add-port=8080/tcp --zone=public

# firewall-cmd --reload

# firewall-cmd --list-all
```

Summary

The `netfilter` subsystem allows kernel modules to inspect every packet traversing the system. All incoming, outgoing or forwarded network packets are inspected. The use of `firewalld` has simplified management by classifying all network traffic into zones. Each zone has its list of ports and services. The public zone is set as the default zone. The `firewalld` service has several pre-defined services. They can be listed using the `firewall-cmd --get-services` command.

Command References:

`firewall-cmd`, `firewalld`

Hostname

The `hostname` command displays or temporarily modifies the system's fully qualified hostname.

```
# hostname
```

Specify a static hostname in the `/etc/hostname` file. Use the `hostnamectl` command to modify this file and may be used to view the status of the system's fully qualified hostname.

```
# hostnamectl set-hostname host.example.com
# hostnamectl status
# cat /etc/hostname
```

The conversion of hostnames to IP addresses or the reverse is determined by the configuration of the `/etc/nsswitch.conf` file. By default, the contents of the `/etc/hosts` file is checked first.

If an entry isn't found in the `/etc/hosts` file, the stub resolver tries to look up the hostname by using a DNS nameserver. `/etc/resolv.conf` file controls how this query is performed.

To test name resolution:

```
# host server.exaple.com
# host 192.168.122.100
```

Summary

The system's static hostname is stored in the `/etc/hostname` file. Use the `hostnamectl` command to modify or view the status of the system's hostname and related settings. The `hostname` command displays or temporarily modifies the system's hostname.

Command References:

`nmcli`, `hostnamectl`, `hosts`, `getent`, `host`, `nsswitch.conf` and `resolv.conf`.

Logout Users

You may need to terminate user sessions and log them off. All user login sessions are associated with a terminal device (`TTY`). Use the `w` command to establish user logins and currently running processes.

```
# w

USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0    192.168.122.1    07:29    3.00s  0.02s  0.00s w
johndoe  pts/1    192.168.122.1    09:08    8.00s  0.00s  0.00s vi sample
```

In this example, the user `johndoe` can be *kicked* off the server using the `pkill` command.

```
# pkill -SIGKILL -t pts/1
```

Summary

User sessions can be terminated by system administrators using the `pkill` command.

Command References:

`pkill`, `pstree`, `signal` and `w`.

Managing Files

All files on a Linux system are stored on a file system. The `/` directory is the root directory at the top of the file-system. Static content remains unchanged until edited or reconfigured. Dynamic or variable content may be modified or appended by active processes, typically kept under `/var`.

Persistent content remains after a reboot, like configuration settings, which might include the `/etc` directory used to keep configuration files.

Runtime content is either process or system-specific content that is deleted by a reboot, found under `/run` for example.

Regular commands and utilities are found in `/usr/bin` whereas installed programs and libraries are under `/usr`.

Command	Description
`mkdir`	Create a directory
`cp`	Copy a file
`cp -r`	Copy a directory and its contents
`mv`	Move or rename a file or directory
`rm`	Remove a file
`rm -r`	Remove a directory containing files
`dir`	List directory contents

It is possible to create multiple names that point to the same file. There are two ways to do this: by creating a hard link to the file, or by creating a soft link (sometimes called a symbolic link) to the file. Each has its advantages and disadvantages.

Every file starts with a single hard link, from its initial name to the data on the file system. When you create a new hard link to a file, you create another name that points to that same data. The new hard link acts exactly like the original file name. Once created, you cannot tell the difference between the new hard link and the original name of the file.

```
# ln existing_sraget_file hard_link_path_name
```

- hard links can only be used with regular files.
- hard links can only be used if both files are on the same file system.

The `ln -s` command creates a soft link, which is also called a "symbolic link." A soft link is not a regular file, but a special type of file that points to an existing file or directory.

Soft links have some advantages over hard links:

- They can link two files on different file systems.
- They can point to a directory or special file, not just a regular file.

Summary

Files on a Linux system are a single, inverted tree of directories, known as a file-system hierarchy. Absolute paths start with a `/` and specify the location of a file in the file-system hierarchy. Relative paths do not start with a `/` and specify the location of a file relative to the current working directory. Hard links and soft links are different ways to have multiple filenames point to the same data.

Command References:

`mkdir`, `rmdir`, `cp`, `mv` and `rm`.

Monitor Processes

The Linux kernel provides a *load average* which is a measurement provided as a rough gauge of how a system is performing and determine load over time.

The load average represents system load over time and determined by reporting how many processes are ready to run on a CPU, and how many processes are waiting for disk or network I/O to complete.

The `uptime` command is one method to display the current load average.

```
# uptime
 09:34:00 up 1 day,  3:13,  2 users,  load average: 0.00, 0.00, 0.00
```

The three values for the load average represent the load over the last one, five, and fifteen minutes.

The `lscpu` command can help you determine how many CPUs a system has.

The `top` command also provides a dynamic real-time view of a running system. Use key `1` while in `top` displays all CPUs on a system.

Summary

There are a few commands that help establish a rough idea of the load on a system.

Command References:

`uptime`, `lscpu` and `top`.

Mounts

The `mount` command allows the root user to mount a file system manually. The first argument of the `mount` command specifies the file system to mount. The second argument specifies the directory to use as the mount point in the file-system hierarchy.

There are two common ways to specify the file system on a disk partition to the `mount` command: With the name of the device file in `/dev` containing the file system. With the `UUID` written to the file system, a universally-unique identifier.

Use the `lsblk` command to list the details of a specified block device or all the available devices. Use the `blkid` to obtain UUID numbers.

```
# mount /dev/sdb1 /data
# mount UUID="d00efaa4-d929-4966-bbda-9b21c7c719a3" /data
```

Unmount a file system with `umount /data`.

To mount an ISO image the `loop` option is required.

```
# mount -o loop /opt/isos/image.iso /opt/iso-mount-point/
```

Use either `parted` or `fdisk` to create new partitions on disks.

```
# parted /dev/sdb
# fdisk /dev/sdb
```

File systems are created using `mkfs.xfs`. To persistently mount a file system upon system boot and entry in `/etc/fstab` needs adding.

Example of the process:

```
# fdisk /dev/sdb
# mkfs.xfs /dev/sdb
# blkid
# vi /etc/fstab
UUID=d00efaa4-d929-4966-bbda-9b21c7c719a3 /data                   xfs     defaults
0 0
```

To force the system to read and mount new entrie in `/etc/fstab` use `systemctl daemon-reload` or `mount -a`.

New disk lebels can be added using `parted`.

```
# parted /dev/sdb mklabel msdos
```

Summary

The mount command allows the root user to manually mount a file system. `fdisk` or `parted` can be used to add, modify, and remove partitions on disks with the MBR or the GPT partitioning scheme. XFS file systems are created on disk partitions using `mkfs.xfs`. To make file system mounts

persistent, they must be added to `/etc/fstab`.

Command References:

`lsblk`, `blkid`, `mount`, `umount`, `parted` and `fstab`.

Networking

Use the `ip` command to inspect network interfaces.

```
# ip a
# ip link show
```

To show statistics about network performance:

```
# ip -s link show enp1s0
```

To test connectivity use the ping command:

```
# ping 192.168.122.10
```

To show the IPv4 routing table:

```
# ip route
```

Use `traceroute` or `tracepath` to trace network traffic takes to reach a remote host through multiple routers:

```
# tracepath www.google.com
```

Use the `ss` command to display socket statistics. The `ss` command is meant to replace the older tool `netstat`:

```
# netstat -an | grep LISTEN
# ss -ta
```

NetworkManager is a daemon that monitors and manages network settings. Command-line and graphical tools talk to NetworkManager and save configuration files under `/etc/sysconfig/network-scripts`.

Use the `nmcli` utility to create and edit connection files from the command line.

```
# nmcli dev status
# nmcli con show
# nmcli con show --active
```

Use the `nmcli con add` command to add new network connections.

```
# nmcli con add con-name enp9s0 type ethernet ifname enp9s0
```

```
# nmcli con add con-name enp10s0 type ethernet ifname enp10s0 ipv6.address
fe80::b608:7af5:c9d3:fa66/64 ipv6.gateway 2001:1:1:1443::400 ipv4.address
192.168.122.120/24 ipv4.gateway 192.168.122.1
```

The `nmcli con up name` command activates the connection, the `nmcli con down name` and the `nmcli dev dis device` to deactivate a network interface.

```
# nmcli con reload
# nmcli con up enp10s0
```

By default, changes made with `nmcli con mod name` are automatically saved to `/etc/sysconfig/network-scripts/ifcfg- name`. It is possible to configure the network by directly editing the connection configuration files.

Example:

```
DEVICE=enp1s0
NAME="enp1s0"
BOOTPROTO=none
IPADDR0=192.168.122.8
PREFIX0=24
GATEWAY0=192.168.122.254
DNS1=192.168.122.254
ONBOOT=yes
```

Summary

The TCP/IP network model is a simplified, four-layered set of abstractions that describes how different protocols interoperate for computers to send traffic from one machine to another over the Internet. NetworkManager is a daemon that monitors and manages network configuration. Use the `nmcli` command for configuring network settings with NetworkManager.

Command References:

`ip`, `ping`, `tracepath`, `traceroute`, `ss`, `netstat`, `NetworkManager`, `nmcli`.

NFS

The Network File System (NFS), is an internet standard protocol used by Linux, UNIX, and similar operating systems as their native network file system.

NFS servers export shares) and NFS clients mount an exported share to a local mount point.

Mount NFS Shares

Temporarily mount an NFS share using the `mount` command:

```
$ sudo mount -t nfs -o rw,sync remoteserver:/share /mount
```

Persistently mount an NFS share using `/etc/fstab`:

```
remoteserver:/share nfs  rw,sync  0 0
```

Use the `nfsconf` tool to manage NFS client and server configuration files to get, set, or unset NFS configuration parameters. It updates the `/etc/nfs.conf` configuration file.

Examples:

```
# nfsconf --set nfsd vers4.2 y
# nfsconf --get nfsd vers4.2
# nfsconf --unset nfsd vers4.2
```

To configure an NFSv4-only client:

```
# nfsconf --set nfsd udp n
# nfsconf --set nfsd vers2 n
# nfsconf --set nfsd vers3 n
```

Automounter

The automounter is a service ` autofs` that automatically mounts NFS shares "on-demand," and automatically unmounts NFS shares when they are no longer used.

Install the autofs package:

```
# dnf install autofs nfs-utils
```

Add a master map file to `/etc/auto.master.d`:

```
# vi /etc/auto.master.d/example.autofs
```

```
/shares  /etc/auto.example
```

Create the mapping file:

```
# vi /etc/auto.example
```

```
*     -rw,hard,intr          192.168.122.200:/nfsdata/&
*     -rw,sync,fstype=nfs4   server.example.com:/shares/&
```

Start the `autofs` service:

```
# systemctl enable  autofs
# systemctl start autofs
```

Example of direct mapping:

```
/-  /etc/auto.direct
```

Summary

Mount and unmount an NFS export from the command line. Configure an NFS export to mount at start-up automatically. Configure the `automounter` with direct and indirect maps, and describe their differences. Configure NFS clients to use NFSv4 using the new `nfsconf` tool.

Command References:

`mount`, `umount`, `fstab`, `nfsconf`, `autofs`.

File Permissions

The command used to change permissions from the command line is `chmod`, which means "change mode" (permissions are also called the mode of a file). The `chmod` command takes a permission instruction followed by a list of files or directories to change. The permission instruction can be issued either symbolically (the symbolic method) or numerically (the numeric method).

Who is `u`, `g`, `o`, `a` (for user, group, other, all)

What is `+`, `-`, `=` (for add, remove, set exactly)

Which is `r`, `w`, `x` (for read, write, execute)

The `chmod` command supports the `-R` option to recursively set permissions on the files in an entire directory tree.

	Owner	Group	All
symbolic	r w x	r w x	r w x
binary	4 2 0	4 2 0	4 2 0
example	1 1 0	1 1 0	1 0 0
decimal	6	6	4

Therefore `664` means `rw-rw-r--`

A newly created file is owned by the user who creates that file. Only root can change the user that owns a file.

```
# chown user test_file
# chown -R user test_dir
# chown :admins test_dir
# chown visitor:guests test_dir
```

Instead of using `chown`, some users change group ownership by using the `chgrp` command. This command works just like `chown`, except only used to change group ownership.

Special permissions constitute a fourth permission type in addition to the primary user, group, and other types. As the name implies, these permissions provide additional access-related features over and above what the basic permission types allow.

The `setuid` permission on an executable file means that commands run as the user owning the file, not as the user that ran the command. One example is the `passwd` command:

```
# ls -l /usr/bin/passwd
```

The special permission `setgid` on a directory means that files created in the directory inherit their group ownership from the directory, rather than inheriting it from the creating user.

```
# ls -ld /run/log/journal
```

A sticky bit for a directory sets a special restriction on the deletion of files. Only the owner of the file (and root) can delete files within the directory. An example is `/tmp`

```
# ls -ld /tmp
```

- Symbolically: setuid = `u+s` ; setgid = `g+s`; sticky = `o+t`
- Numerically (fourth preceding digit): setuid = `4`; setgid = `2`; sticky = `1`

The `umask` command without arguments displays the current value of the shell's umask:

The system's default umask values for Bash shell users are defined in the `/etc/profile` and

`/etc/bashrc` files. Users can override the system defaults in the `.bash_profile` and `.bashrc` files in their home directories.

As `root` , you can change this by adding a shell startup script named `/etc/profile.d/local-umask.sh`.

```
#!/bin/bash
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 007
else
    umask 022
fi
```

Summary

Files have three categories to which permissions apply. A file is owned by a user, a single group, and other users. The most specific permission applies. User permissions override group permissions and group permissions override other permissions.

The `chmod` command changes file permissions from the command line. There are two methods to represent permissions, symbolic (letters) and numeric (digits).

The `chown` command changes file ownership. The `-R ` option recursively changes the ownership of a directory tree.

The `umask` command without arguments displays the current umask value of the shell and the default umask values for Bash are defined in the `/etc/profile` and `/etc/bashrc` files.

Command References:

`chmod`, `chown`, `chgrp`, `ls`, `chmod` and `umask`.

Signals

Signals are software interrupts sent to a program to indicate that an event has occurred. Events that generate a signal can be an error, external event, or by use of a signal-sending command.

The `kill` command sends a signal to a process by PID number. Despite its name, the `kill` command sends any signal, not just those for terminating programs. You can use the `kill -l` command to list the names and numbers of all available signals.

```
 1) SIGHUP       11) SIGSEGV
 2) SIGINT       12) SIGUSR2
 3) SIGQUIT      13) SIGPIPE
 4) SIGILL       14) SIGALRM
 5) SIGTRAP      15) SIGTERM
 6) SIGABRT      16) SIGSTKFLT
 7) SIGBUS       17) SIGCHLD
 8) SIGFPE       18) SIGCONT
 9) SIGKILL      19) SIGSTOP
10) SIGUSR1      20) SIGTSTP
```

System administrators, however, are most familiar with number `9` the SIGKILL, which causes abrupt program termination.

Use the `ps` command to view running process IDs (PIDs) and then the `kill` command to terminate it.

```
# ps -ef | grep firewall

root       771  ...output omitted...
```

```
# kill -9 771
```

Summary

A signal is a software interrupt that reports events to an executing program. The `kill` , `pkill` and `killall` commands use signals to control processes.

Command References:

`kill`, `killall`, `ps` and `pgrep`.

Process Priority

Linux systems run more processes than there are processing units. A technique called time-slicing or multitasking is used by the operating system process scheduler to rapidly switch between processes on a single-core, giving the impression that multiple processes are running at the same time.

```
# ps axo pid,comm,nice,cls --sort=-nice
```

Not all processes are equally important. Processes running with the SCHED_NORMAL policy can be given a relative priority. This priority is called the nice value of a process. The `nice` level values range from -20 (highest priority) to 19 (lowest priority). By default, processes inherit their `nice` level from their parent, which is usually 0.

Without options, the `nice` command starts a process with the default nice value of 10.

```
nice -n 15 command &
renice -n 19 1234
```

Summary

A relative priority is assigned to a process to determine its CPU access. This priority is called the `nice` value of a process. The `nice` command assigns a priority to a process when it starts. The `renice` command modifies the priority of a running process.

Command References:

`nice`, `renice` and `top`.