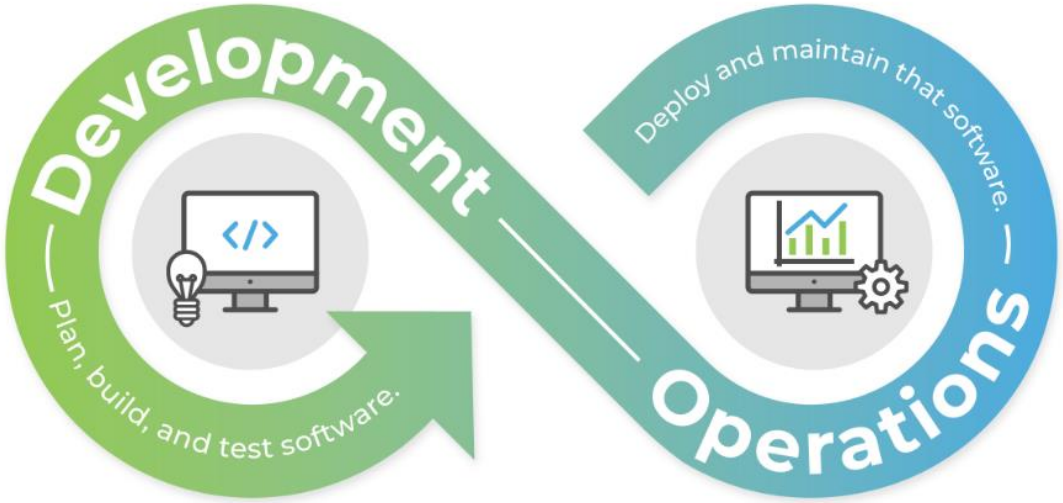
DevOps
01 Linux Basics

Linux Commands & More - Part 2



Agenda for this session

- Redirection & Filters, Simple Filter and Advance Filter commands
- Start and Stop Services
- Find & Kill a process with id and name
- Package installation using RPM and YUM

Redirection and Filter Commands (Simple and Advanced)

Redirecting Standard Output to a File (Using ‘>’, ‘>>’)

`ls -la > mylist`	creates or overwrites file
`ls -la >> mylist`	appends or creates file

Redirecting Standard Input from a File (Using ‘<’)

`cat < mylist` passes the file ‘mylist’ to ‘cat’ command

Filter Commands

`cat sample.txt`	lists all lines in sample.txt
`head sample.txt`	lists the first 10 lines of sample.txt
`head -5 sample.txt`	lists the first 5 lines of sample.txt
`tail sample.txt`	lists the last 10 lines of sample.txt
`tail -5 sample.txt`	lists the last 5 lines of sample.txt

Filter Commands

`sort sample.txt`	sorts and lists all lines in sample.txt
`uniq sample.txt`	removes adjacent duplicate lines and lists sample.txt
`sort sample.txt | uniq`	lists unique lines of sample.txt in alphabetical order
`wc sample.txt`	lists the number of lines, words and characters in sample.txt
`tac sample.txt`	lists the lines of sample.txt in reverse order (opposite of cat)

Filter Commands

`grep mit sample.txt` lists all lines in sample.txt having the string ‘mit’

`sed ‘s/mit/MIT/g’ sample.txt` replaces ‘mit’ with ‘MIT’ in all places in sample.txt

`nl sample.txt` lists sample.txt with a serial number starting with 1 per line

Try this – Create student.dat with the following lines (use vi or vim or any editor)

```
001 Amit Sahu
002 Cyndrella Murphy
003 Dhruv Shah
004 Manish Kulkarni
005 Balaji Joshi
001 Amit Sahu
004 Manish Kulkarni
002 Cyndrella Murphy
003 Dhruv Shah
005 Balaji Joshi
004 Manish Kulkarni
002 Cyndrella Murphy
```

Try the following commands

```
cat student.dat
sort student.dat
sort +0 student.dat
sort +1 student.dat
sort +2 student.dat
sort +3 student.dat (What is the output? Why?)
uniq student.dat
cat student.dat | uniq
cat student.doc | sort | uniq
sort student.doc | uniq | wc -l
echo student.dat file has $(cat student.dat|sort|uniq|wc -l unique
lines
```

Redirecting Standard Output to a Command (Using Pipe '|')

`ls –alr | more` passes the output of 'ls –alr' to 'more' command

`cat x.doc | wc –l` passes the output of 'cat x.doc' to 'wc -l' command and displays the number of lines in x doc.

```
echo x.doc has $(cat x.doc | wc -l) lines
```

assuming that there are 10 lines in x.doc, it displays

```
x.doc has 10 lines
```

Combination of Pipes and Redirection

```
cat x.doc | wc -l > output.txt

cat output.txt

10
```

Standard Input (STDIN), Standard Output (STDOUT), and Standard Error (STDERR)

By default,

STDIN is the keyboard (you type a command and it appears on the terminal as you type)

STDOUT is the terminal window (CLI)

STDERR is the terminal window (CLI)

These three locations are file descriptors. File descriptors are used to represent a file used by a process.

STDIN	0
STDOUT	1
STDERR	2

FILE DESCRIPTORS

Example

```
find / -iname '*mit*'
```

This command is to find from '/' all files and directories with name '*mit*' whether the file name is in smaller case or upper case. Note: '-iname' indicates case insensitive operation and '-name' indicates case sensitive operation.

Typical output of this command could be,

```
/usr/share/doc/mit_intro.doc
/usr/share/doc/something/examples/events_at_Mit.doc
find: `/run/udisks2': Permission denied
find: `/run/wpa_supp': Permission denied
/usr/share/clubs_MIT_progress.txt
/usr/games/mit_game1
```

We are getting these two error messages because the user who runs this command does not have access to these two files or directories. These error messages are written to STDERR.

To hide **stderr**, we can redirect them by referencing standard error's file descriptor number, **2**, and a "redirect output" operator, **>**.

Example (Contd..)

```
find / -iname '*mit*' 2> errors.txt
```

This command is to find from '/' all files and directories with name '*mit*' whether the file name is in smaller case or upper case. Error messages will be recorded in a file 'errors.txt'.

```
/usr/share/doc/mit_intro.doc
/usr/share/doc/something/examples/events_at_Mit.doc
/usr/share/clubs_MIT_progress.txt
/usr/games/mit_game1
```

```
cat errors.txt
```

This command displays the contents of errors.txt on STDOUT (terminal screen)

```
find: `/run/udisks2': Permission denied
find: `/run/wpa_supp': Permission denied
```

Commands (Text Processing)

awk	Pattern scanning and processing language
cat	Display file(s)
cut	Extract selected fields of each line of a file
diff	Compare two files
grep	Search text for a pattern
head	Display the first 10 lines of a file
less	Display files on a page-by-page basis
sed	Stream editor (esp. search and replace)
sort	Sort text files
split	Split files
tail	Display the last 10 lines of a file
tr	Translate/delete characters
uniq	Filter out repeated lines in a file
wc	Line, word and character count

printenv command

<u>Try the following</u>:

`printenv > env_variables` (creates a file called env_variables)

`cat env_variables`

`wc -l env_variables`

`printenv | wc -l`

`printenv | head` shows the first 10 lines from the output of printenv

`printenv | tail` shows the last 10 lines from the output of printenv

`printenv | more` shows page by page view of the output

Services – Starting and Stopping

What is a Service in Linux?

A service is a piece of operating system code used to handle some type of request.

System administrators are responsible for working with services (starting or stopping them).

A service is also known as daemon (pronounced ‘demon’).

Categories of Linux Services

The Linux services can be broken down into several different categories such as,

- Boot
- File system
- Hardware
- Language support
- Logging
- Network, web/Internet
- Power management
- Scheduling
- System maintenance

Some services support a range of requests while other services must combine to perform their task.

Features or characteristics of a service

1. It runs in the background so that it <u>does not take up processor time unless called upon</u>
2. Services can handle requests that come from many different sources: users, applications software, hardware, other operating system services, messages from the network
3. Services are configurable. Configuration is usually handled through configuration files, which are often stored in the /etc directory (or a subdirectory).
4. Services can be running or stopped, and the system administrator can control which services are running or stopped and which services are automatically started at system initialization time based on the runlevel.

Services can be managed in two ways

1. Using systemd
2. Using init

systemd

- systemd is a Linux initialization system and service manager
- It includes features like on-demand starting of services (also known as daemons), mount and automount point maintenance, snapshot support, and processes tracking using Linux control groups.
- systemd provides a logging daemon and other tools and utilities to help with common system administration tasks.
- In systemd, we can manage services with systemctl command.

Read: https://www.linode.com/docs/guides/what-is-systemd/

Listing all services that are ‘running’

```
systemctl | grep running
```

```
ubuntu@ubuntu-VirtualBox: ~
File  Edit  View  Search  Terminal  Help
ubuntu@ubuntu-VirtualBox:~$ sudo systemctl | grep running
[sudo] password for ubuntu:
acpid.path                                                          loaded active running   ACPI Events Check
cups.path                                                           loaded active running   CUPS Scheduler
init.scope                                                          loaded active running   System and Service Manager
session-1.scope                                                     loaded active running   Session 1 of user ubuntu
accounts-daemon.service                                             loaded active running   Accounts Service
acpid.service                                                       loaded active running   ACPI event daemon
avahi-daemon.service                                                loaded active running   Avahi mDNS/DNS-SD Stack
bolt.service                                                        loaded active running   Thunderbolt system service
clean-mount-point@media-ubuntu-VBox_GAs_6.0.4.service               loaded active running   Clean the /media/ubuntu/VBox_GAs_6.0.4 mount point
colord.service                                                      loaded active running   Manage, Install and Generate Color Profiles
cron.service                                                        loaded active running   Regular background program processing daemon
cups-browsed.service                                                loaded active running   Make remote CUPS printers available locally
cups.service                                                        loaded active running   CUPS Scheduler
dbus.service                                                        loaded active running   D-Bus System Message Bus
fwupd.service                                                       loaded active running   Firmware update daemon
gdm.service                                                         loaded active running   GNOME Display Manager
```

Sample output of the command

Source: https://itsfoss.com/start-stop-restart-services-linux/

systemd - Start, Stop, Restart, & Check Status

```
systemctl start <service_name>

systemctl stop <service_name>

systemctl restart <service_name>

systemctl status <service_name>
```

Another way is to use init

- It is the first process executed by the kernel. It is a daemon process which runs till the system is shutdown. That is why, <u>it is the parent of all the processes</u>.
- init reads the script stored in the file **/etc/inittab** to take care of everything that a system do at the time of system initialization like setting the clock, initializing the serial port and so on.

Read: https://www.javatpoint.com/linux-init

Managing services with init

```
service <service_name> start

service <service_name> stop

service <service_name> restart

service <service_name> status
```

Finding and killing processes with id and name

Processes

Process is a running program. Processes can start other processes. When we interact with Linux, we create numbered instances of running programs called “processes.”

`ps`	lists processes
`ps -ef`	lists all processes running in the system

To get manual pages on ps command, try

```
man ps
```

More on processes

Linux assigns a process id (PID) to every process that is started. Whenever a process runs, Linux keeps track of it through PID.

After booting, the first process is an initialization process called init. It is given a PID of 1. From that point on, each new process gets the next available PID.

If your Linux system has been running for a while, you might find PIDs that are large (4 or 5 digits or even more).

Parent and Child Processes

A process can only be created by another process.

We refer to the creating process as the parent and the created process as the child.

The parent process spawns one or more child processes.

The spawning of a process can be accomplished in one of several ways. Each requires a system call (function call) to the Linux kernel. These function calls are fork(), vfork(), clone(), wait(), and exec().

<u>Question:</u> What is a ‘Zombie’ process? (Hint: Read selected pages from the book ‘Linux with Operating System Concepts’ by Richard Fox

Active processes

`top` displays active processes (‘q’ to quit)

Use up and down arrow keys to scroll up and down the list of processes.

Windows PowerShell | raja@LAPTOP-FV4AF8PS: ~/d

```
top - 12:13:19 up  2:43,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:  34 total,   1 running,  33 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3817.9 total,   2575.3 free,    526.3 used,    716.4 buff/cache
MiB Swap:   1024.0 total,   1024.0 free,      0.0 used.   3087.1 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
      1 root      20   0  167036  12456   8212 S   0.0   0.3   0:00.84 systemd
      2 root      20   0    2324   1196   1084 S   0.0   0.0   0:00.00 init-systemd(Ub
      5 root      20   0    2352     72     68 S   0.0   0.0   0:00.00 init
     38 root      19  -1   47728  14676  13668 S   0.0   0.4   0:00.10 systemd-journal
     63 root      20   0   21956   5912   4460 S   0.0   0.2   0:00.22 systemd-udevd
     79 root      20   0    4496    184     36 S   0.0   0.0   0:00.00 snapfuse
     80 root      20   0    4496    208     56 S   0.0   0.0   0:00.00 snapfuse
     82 root      20   0    4628    180     32 S   0.0   0.0   0:00.00 snapfuse
     83 root      20   0    4688   1652   1180 S   0.0   0.0   0:01.06 snapfuse
     86 root      20   0    4496    160     12 S   0.0   0.0   0:00.00 snapfuse
     88 root      20   0    4784   1828   1268 S   0.0   0.0   0:02.86 snapfuse
     96 root      20   0    4960   1720   1232 S   0.0   0.0   0:01.12 snapfuse
    105 systemd+  20   0   25528  12220   8024 S   0.0   0.3   0:00.09 systemd-resolve
    121 root      20   0    4304   2700   2464 S   0.0   0.1   0:00.01 cron
    123 message+  20   0    8588   4588   4052 S   0.0   0.1   0:00.12 dbus-daemon
    127 root      20   0   30124  19152  10340 S   0.0   0.5   0:00.08 networkd-dispat
    128 syslog    20   0  222400   7348   4532 S   0.0   0.2   0:00.01 rsyslogd
    129 root      20   0 1540472  58392  19948 S   0.0   1.5   0:03.42 snapd
    130 root      20   0   15320   7212   6272 S   0.0   0.2   0:00.10 systemd-logind
    189 root      20   0  107220  21628  13080 S   0.0   0.6   0:00.06 unattended-upgr
    221 root      20   0    3236   1048    956 S   0.0   0.0   0:00.00 agetty
    223 root      20   0    3192   1100   1004 S   0.0   0.0   0:00.00 agetty
```

Creating a shell program ‘countdown.sh’

Use an editor (vi or vim or nano). Add the following lines.

```
clear
echo "Coundown Program Started on " $(date)
declare j num
j=11
for ((i=1; i<=10; i++))
do
        sleep 2
        echo Waiting $(($j-$i)) seconds
done
echo "Countdown Program Ended on " $(date)
```

Make countdown.sh executable by running the command ‘chmod +x countdown.sh’

Running the shell program ‘countdown.sh’

Type ./countdown.sh at the $ prompt.

```
$ ./countdown.sh
```

Wait for the $ prompt. Now, redirect the output to out.txt. For this use the following command. Observe what happens.

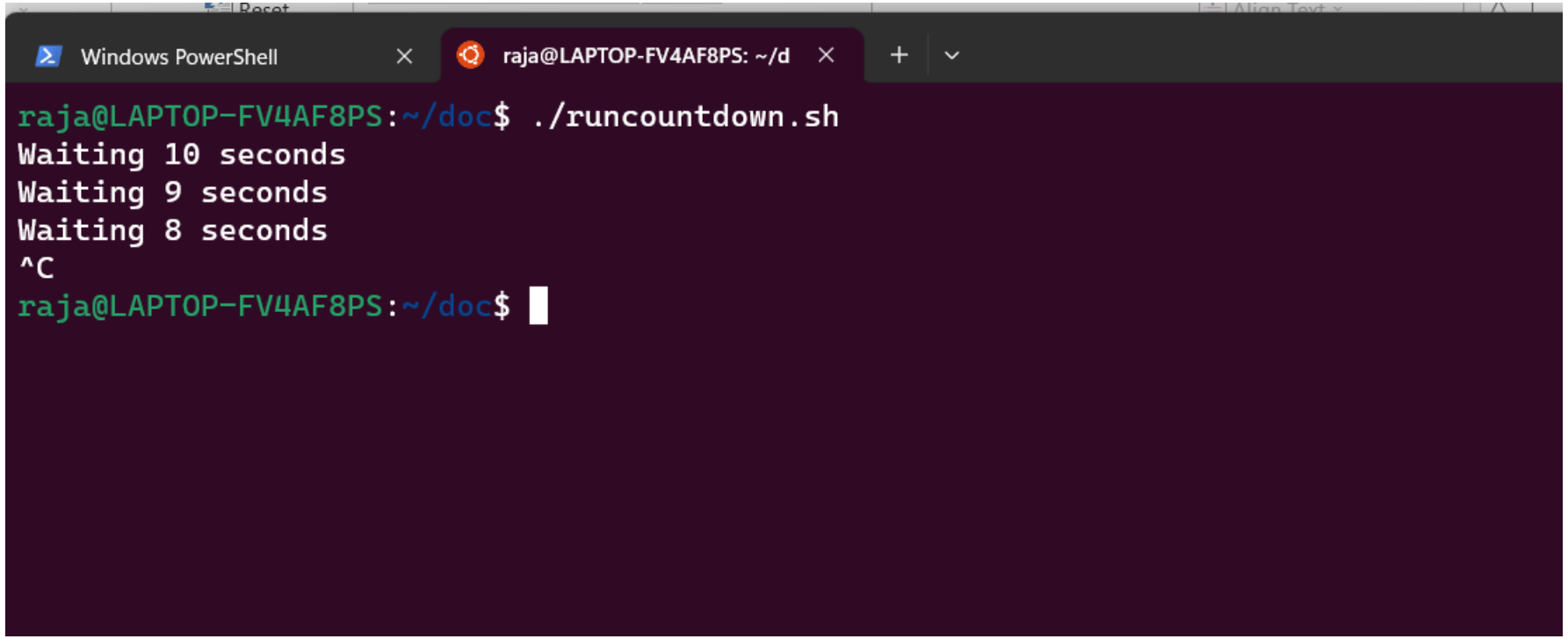
```
$ ./countdown.sh > out.txt
```

Use ‘`cat out.txt`’ to see the content of the file. Next, redirect the output to out.txt and run it as a background process. For this, use the following command. (& is used to run a command in background)

```
$ ./countdown.sh > out.txt &
```

Killing a process with <ctrl>C

When you start a process (run a long program for example), you can stop it using <ctrl>C



Kill command (to kill a process)

& is used to initiate a background process

```
raja@LAPTOP-FV4AF8PS:~/doc$ ./runcountdown.sh > out.txt &
[1] 507
raja@LAPTOP-FV4AF8PS:~/doc$ ps -la
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000     398     349  0  80   0 -  1530 core_s pts/1    00:00:00 bash
0 S  1000     507     347  0  80   0 -  1228 do_wai pts/0    00:00:00 runcountdown.sh
0 S  1000     510     507  0  80   0 -   802 hrtime pts/0    00:00:00 sleep
0 R  1000     511     347  0  80   0 -  1870 -      pts/0    00:00:00 ps
raja@LAPTOP-FV4AF8PS:~/doc$ kill 507
raja@LAPTOP-FV4AF8PS:~/doc$ ps -la
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000     398     349  0  80   0 -  1530 core_s pts/1    00:00:00 bash
0 R  1000     517     347  0  80   0 -  1870 -      pts/0    00:00:00 ps
[1]+  Terminated              ./runcountdown.sh > out.txt
raja@LAPTOP-FV4AF8PS:~/doc$
```

507 is the process id of this running program

Killing a process immediately

-9 option can be used in kill command to send a signal that can terminate any process immediately when attached with a PID or a process name. It is a forceful way to kill or terminate a process or set of processes.

kill -9 <pid> / <processname>

```
kill -9 507
```

This command sends SIGKILL (9) — kill signal. This signal cannot be handled (caught), ignored or blocked.

Read

- https://itsfoss.com/how-to-find-the-process-id-of-a-program-and-kill-it-quick-tip/
- https://itsfoss.com/kill-zombie-process-linux/
- https://www.javatpoint.com/kill-process-linux
- https://www.tecmint.com/find-and-kill-running-processes-pid-in-linux/
- https://stackoverflow.com/questions/3510673/find-and-kill-a-process-in-one-line-using-bash-and-regex
- Chapter 4 of “Linux with Operating System Concepts” by Richard Fox

Services and Processes in Linux

One of the most powerful features of Linux is its ability to manage processes and services.

Processes are individual instances of a program that run on the system, while services are background processes that provide various functions to the system.

Package installation using RPM and YUM

What is a package

It is a compressed software archive file containing all the files included with a software application (pre-compiled binary software files, installation scripts, configuration files, dependency requirements, and other details about the software).

It can be a command-line utility, GUI application, or software library.

Most software applications of Linux are distributed as packages.

Read: https://www.scaler.com/topics/cyber-security/package-management-in-linux/

Types of packages

1. RPM packages (. rpm)
2. Debian packages (. deb)
3. TAR archives (. tar)
4. TGZ archives (. tgz)
5. GZip Archives (. gz)

Read: https://www.cbtnuggets.com/blog/certifications/open-source/the-5-linux-packaging-types-you-need-to-know

Linux administrators must be familiar with the various types of packaging formats that are used with Linux.

What is package management

Package management is a method of installing, updating, removing, and keeping track of software updates from specific repositories (repos) in the Linux system.

Linux distros often use different package management tools.

Ubuntu, Fedora, RedHat are examples of Linux distros

(The short form of ‘distributions’ is ‘distros’).

Repositories

We use software repositories (also called repos) to obtain packages. Repositories are simply the location where the packages are stored, commonly accessible through the internet.

A repository can contain a single package or thousands of packages.

Most Linux distributions have their own unique repositories.

Dependencies

In Linux, each package contains metadata about the additional packages that are required. These additional packages are called dependencies.

A single package can sometimes have hundreds of dependencies. When installing, upgrading, or removing packages, these dependencies may also need to installed, upgraded, and optionally removed.

Package Manager

A package manager is used to get (access and download), install, upgrade, and remove packages and their dependencies.

It reduces the complexity and improves ‘ease of use’ because it automates the process of obtaining, installing, upgrading, and removing packages and their dependencies.

This dramatically improves the user experience and the ability to properly and efficiently manage software on your Linux system.

Package Manager - Advantages

1. **Ease of getting what you need** - Easily obtain the correct, trusted, and stable package for your Linux distribution.
2. **Automatic dependency management** - Automatically manage all dependencies when installing/updating/uninstalling a package.
3. **Standardized approach** - Linux distributions have conventions or standards regarding how applications are configured and stored in the /etc/ and /etc/init.d/ directories. By using packages, distributions are able to enforce a single standard.
4. **Ease of applying patches and security upgrades** – A single command to automatically update all packages to the latest versions stored on the configured repositories.

Read: https://www.linode.com/docs/guides/linux-package-management-overview/

RPM

RPM (Red Hat Package Manager) is a command-line utility for installing software packages from an RPM file. Also, it helps in updating or removing software packages.

RPM packages contain software programs and libraries, as well as information about the dependencies required to run those programs.

Key Features:

1. Dependency Resolution: Resolves dependencies during installation or update.
2. Rollback Support: Allows users to rollback to a previous version of a package if needed.
3. Verification: Verifies package integrity to ensure that there is no tampering.
4. Scripting Support: Allows users to run scripts during package installation or removal.
5. Source Code Management: Can be used to build and manage source code packages.

RPM – Modes for package management

Every mode has its own set of options. Use ‘man rpm’ command to know more.

MODE	DESCRIPTION
-i	Installs a package
-U	Upgrades a package
-e	Erases a package
-V	Verifies a package
-q	Queries a package

GENERAL OPTIONS	PURPOSE
--version	Prints version number
-v	Prints verbose output
--help	Prints help

COMMAND EXAMPLES:

```
rpm -i package-file
rpm -U package-file
rpm -iv package-file
```

YUM

YUM (Yellowdog Updater Modified) is a package management system used in Linux distributions like Fedora, CentOS, and Red Hat Enterprise Linux. YUM is a command-line tool. It is a front-end to the RPM package manager and provides a user-friendly interface for managing software packages. It uses a repository-based system, where software packages are downloaded from remote repositories and installed on the local system.

Key Features:

1. Dependency Resolution: Resolves dependencies during installation or update.
2. Rollback Support: Allows users to rollback to a previous version of a package if needed.
3. Automatic Updates: Can automatically update packages to their latest available version.
4. Plugin Support: Allows users to extend its functionality using plugins.
5. Repository Management: Helps manage remote repositories and configure their settings.

Read: https://sysadminxpert.com/difference-between-yum-and-rpm/

YUM

COMMAND	PURPOSE
yum install	Installs the specified packages
remove	Removes the specified packages
search	Searches package metadata for keywords
info	Lists description
update	Updates each package to the latest version
repolist	Lists repositories
history	Displays history

COMMAND FORMAT

```
yum -option command
```

OPTIONS	PURPOSE
-C	Runs from system cache
--security	Includes packages that provide a fix for a security issue
-y	Answers yes to all questions
--skip-broken	Skips packages causing problems
-v	Verbose

COMMAND EXAMPLES

```
yum install firebox
yum remove firebox
yum update firebox
yum list installed
```

References

- https://www.javatpoint.com/linux-package-manager
- https://www.javatpoint.com/rpm-command-in-linux
- https://www.redhat.com/sysadmin/how-manage-packages
- https://www.tecmint.com/20-practical-examples-of-rpm-commands-in-linux/
- https://www.tecmint.com/20-linux-yum-yellowdog-updater-modified-commands-for-package-mangement/
- How to Install VIM Editor on Linux (RHEL / CentOS 7/8) Using 6 Easy Steps | CyberITHub
- https://en.wikipedia.org/wiki/Yum_(software)
- https://en.wikipedia.org/wiki/RPM_Package_Manager
- https://phoenixnap.com/kb/rpm-vs-yum

Summary

- Redirection & Filters, Simple Filter and Advance Filter commands
- Start and Stop Services
- Find & Kill a process with id and name
- Package installation using RPM and YUM

Any Questions?

Git installation

```
[sudo] password for raja:
sudo: 2 incorrect password attempts
raja@LAPTOP-FV4AF8PS:~$ sudo passwd root
[sudo] password for raja:
New password:
Retype new password:
passwd: password updated successfully
raja@LAPTOP-FV4AF8PS:~$ sudo apt-get install git-core
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'git' instead of 'git-core'
git is already the newest version (1:2.34.1-1ubuntu1.9).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 22 not upgraded.
raja@LAPTOP-FV4AF8PS:~$
```

This is to set sudo password

This is to install git

Signing in as superuser using su

```
raja@LAPTOP-FV4AF8PS:~$ pwd
/home/raja
raja@LAPTOP-FV4AF8PS:~$ whoami
raja
raja@LAPTOP-FV4AF8PS:~$ su
Password:
root@LAPTOP-FV4AF8PS:/home/raja# whoami
root
root@LAPTOP-FV4AF8PS:/home/raja# pwd
/home/raja
root@LAPTOP-FV4AF8PS:/home/raja# exit
exit
raja@LAPTOP-FV4AF8PS:~$ whoami
raja
raja@LAPTOP-FV4AF8PS:~$
```

Thank You