Tutorial Link https://course.testpad.chitkara.edu.in/tutorials/Advanced Linux Concepts/63d768a027750352bc2e8f7d

TUTORIAL

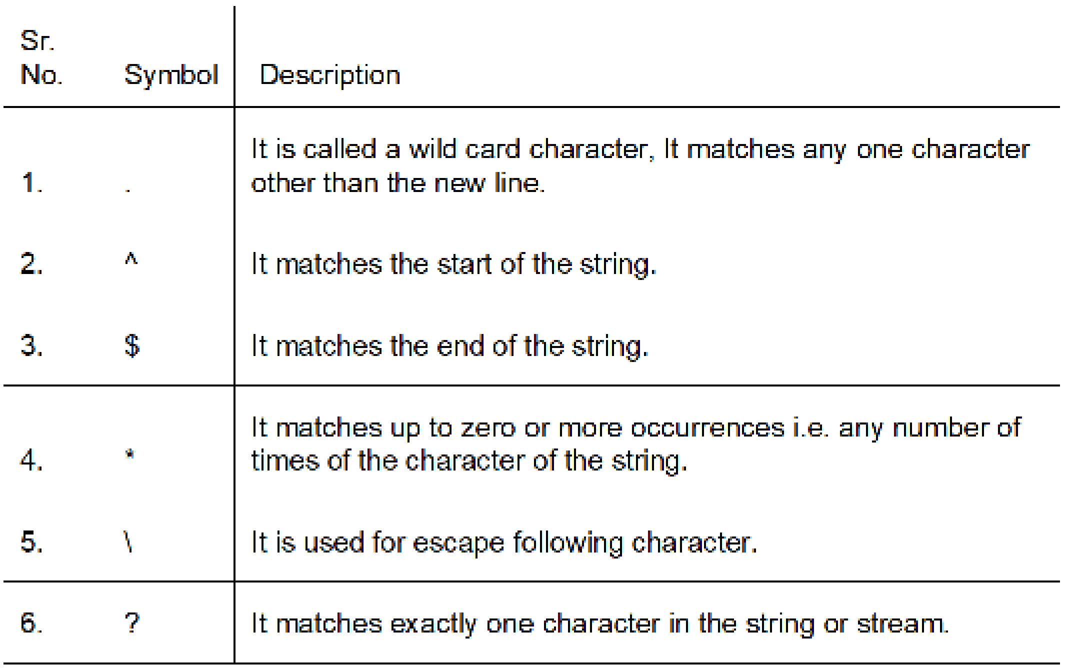
Advanced Linux Concepts

Topics

Regular Expressions

Regular expressions are special characters or sets of characters that help us to search for data and match the complex pattern. Regexps are most commonly used with the Linux commands:- grep, sed, tr, vi.

The following are some basic regular expressions:



Sr. No.	Symbol	Description
1.	.	It is called a wild card character, It matches any one character other than the new line.
2.	^	It matches the start of the string.
3.	$	It matches the end of the string.
4.	*	It matches up to zero or more occurrences i.e. any number of times of the character of the string.
5.	\	It is used for escape following character.
6.	?	It matches exactly one character in the string or stream.

1. Using "." (dot) to match strings.

If there is a file that have following contents:

Apple

Orange

Tomato

Now if we run below command, output will be apple.

```
# cat fruit.txt | grep App.e
```

2. Using "^" (caret) to match the beginning of the string

```
# cat fruit.txt | grep ^A
```

3. Using "$" (dollar sign) to match the ending of the string

```
# cat fruit.txt | grep o$
```

Output will be Tomato

4. Using "*" (an asterisk) to find any number of repetitions of a string

Using "*", we can match up to zero or more occurrences of the character of the string.

```
# cat fruit.txt | grep *.doc
```

It will Display all the file whose extension is .doc

5. Using "\" (a backslash) to match the special symbol

Using "\" with special symbols like whitespace (" "), newline("\n"), we can find strings from the file.Eg to display spaces in between full name of a fruit.

```
# cat fruit.txt | grep "\ "`
```

6. Using "?"(question mark) to find all the matching characters

Using "?", we can match 0 or 1 repetitions of the preceding. For example, if we do something like this: ab? It will match either 'a' or 'ab'.

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. Red Hat-based distros use `RPM` (RPM Package Manager) and `YUM/DNF` (Yellow Dog Updater, Modified/Dandified YUM).

Yellow Dog Updater, Modified (YUM)

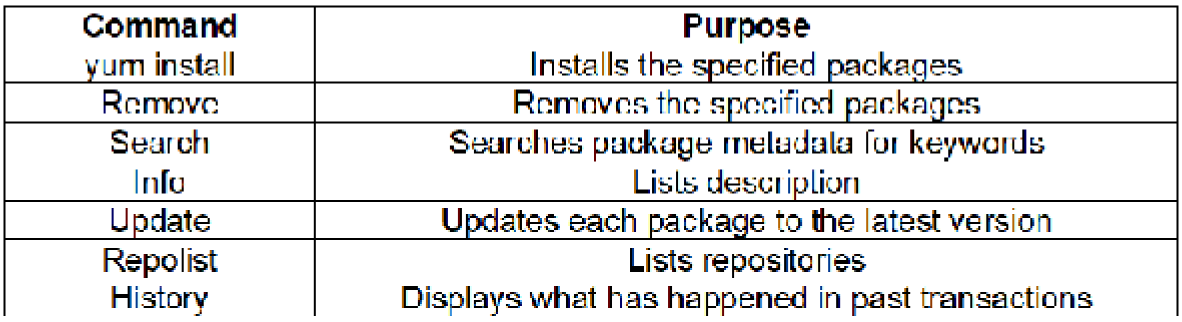
DNF or Dandified YUM is the updated default since Red Hat Enterprise Linux 8, CentOS 8, Fedora 22, and any distros based on these. Generally, the options are the same.

`YUM` is the primary package management tool for installing, updating, removing, and managing software packages in Red Hat Enterprise Linux. `YUM` performs dependency resolution when installing, updating, and removing software packages. `YUM` can manage packages from installed repositories in the system or from `.rpm` packages. The main configuration file for `YUM` is at `/etc/yum.conf`, and all the repos are at `/etc/yum.repos.d`.

It's easy to manage packages in Linux with `YUM`. At the command line, enter:

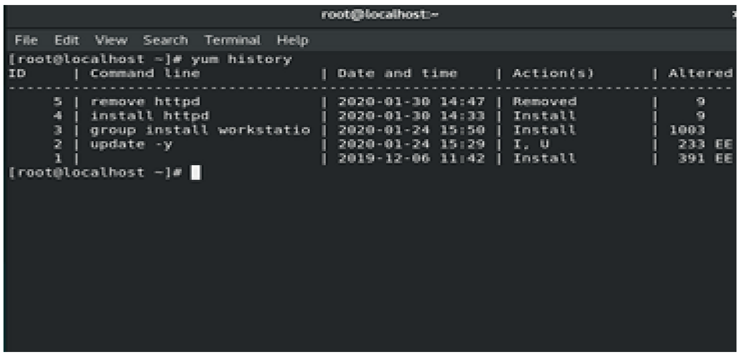
```
# yum -option command
```

There are many options and commands available to use with `YUM`. I've listed some commonly-used commands for `YUM` below:



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 what has happened in past transactions

The `history` option gives you an overview of what happened in past transactions. This provides some useful information, like the date when the transaction happened and what command was run.



```
[root@localhost ~]# yum history
ID     | Command line             | Date and time    | Action(s)      | Altered
-------------------------------------------------------------------------------
     5 | remove httpd             | 2020-01-30 14:47 | Removed        |    9
     4 | install httpd            | 2020-01-30 14:33 | Install        |    9
     3 | group install workstatio | 2020-01-24 15:50 | Install        | 1003
     2 | update -y                | 2020-01-24 15:29 | I, U           |  233 EE
     1 |                          | 2019-12-06 11:42 | Install        |  391 EE
[root@localhost ~]#
```

You can undo or redo certain transactions using the `history` command. Here is an example of undoing a transaction:

```
yum history undo <id>
```

`YUM` provides many options for package management. For detailed option information, look at `man yum` and `yum -help`.

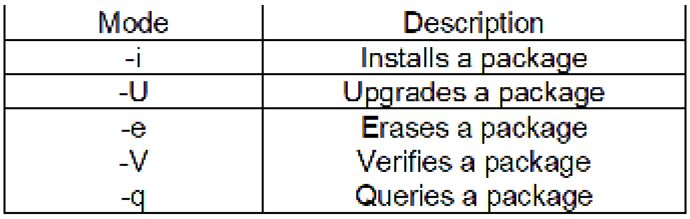
RPM (RPM Package Manager)

`RPM` is a popular package management tool in Red Hat Enterprise Linux-based distros. Using `RPM`, you can install, uninstall, and query individual software packages. Still, it cannot manage dependency resolution like `YUM`. `RPM` does provide you useful output, including a list of required packages. An `RPM` package consists of an archive of files and metadata. Metadata includes helper scripts, file attributes, and information about packages.

`RPM` maintains a database of installed packages, which enables powerful and fast queries. The `RPM` database is inside `/var/lib`, and the file is named `__db*`.

RPM has some basic modes: query, verify, install, upgrade, erase, show querytags, show configuration. At least one of these modes needs to be selected to perform package management tasks. Every mode has its own set of options. For example, install mode `i` has its own set of installation options. Options for the modes are found on the RPM man pages at `man rpm`.

Some commonly used modes are listed below:



Mode	Description
-i	Installs a package
-U	Upgrades a package
-e	Erases a package
-V	Verifies a package
-q	Queries a package

Example: if you want to install/upgrade a package “package-file”:

```
# rpm -i package-file

# rpm -U package-file

# rpm -ivh package-file
```

Kernel management and compilation

The kernel is a core part of a Linux operating system that manages the system resources and provides interface between hardware and software applications. The Red Hat kernel is a custom-built kernel based on the upstream Linux mainline kernel that Red Hat engineers further develop and harden with a focus on stability and compatibility with the latest technologies and hardware.

The Linux kernel RPM package

The `kernel` RPM is a meta package that does not contain any files, but rather ensures that the following required sub-packages are properly installed:

· `kernel-core` - contains the binary image of the kernel, all initramfs-related objects to bootstrap the system, and a minimal number of kernel modules to ensure core functionality. This sub-package alone could be used in virtualized and cloud environments to provide a Red Hat Enterprise Linux 9 kernel with a quick boot time and a small disk size footprint.

· `kernel-modules` - contains the remaining kernel modules that are not present in `kernel-core`.

The small set of `kernel` sub-packages above aims to provide a reduced maintenance surface to system administrators especially in virtualized and cloud environments.

Optional kernel packages are for example:

· `kernel-modules-extra` - contains kernel modules for rare hardware and modules which loading is disabled by default.

· `kernel-debug` — contains a kernel with numerous debugging options enabled for kernel diagnosis, at the expense of reduced performance.

· `kernel-tools` — contains tools for manipulating the Linux kernel and supporting documentation.

· `kernel-devel` — contains the kernel headers and makefiles sufficient to build modules against the `kernel` package.

· `kernel-abi-stablelists` — contains information pertaining to the RHEL kernel ABI, including a list of kernel symbols that are needed by external Linux kernel modules and

`dnf` plug-in to aid enforcement.

· `kernel-headers` — includes the C header files that specify the interface between the Linux kernel and user-space libraries and programs. The header files define structures and constants that are needed for building most standard programs.

Updating the kernel

1. To update the kernel, enter the following command:

```
# dnf update kernel
```

This command updates the kernel along with all dependencies to the latest available version.

2. Reboot your system for the changes to take effect.

3. To install a specific kernel version, enter the following command:

```
# dnf install kernel-{version}
```

4. List the installed kernels using the following command:

```
  # grubby --info=ALL | grep title
```

5. To list all the loaded kernel Modules, enter below commands:

```
 $ lsmod
```

6. To display information about any kernel module, enter:

```
$ modinfo <KERNEL_MODULE_NAME>
```