Linux Training 3

Prepared by Enes Erten

Contents

- Packaging overview
- Package Management with RPM
- Advanced Package Management Concept and yum
- Regex
- grep, egrep, fgrep

Packaging overview

- Packages and Packaging: A software package is a group of files organized in a directory structure along with metadata and intelligence that make up a software application. They are available in two types: binary (or installable) and source.
- Binary packages are installation-ready and are bundled for distribution.
- Source packages come with the original unmodified version of the software that may be unpacked, modified as desired, and repackaged in the binary format for installation or redistribution. They are identified with the .src extension.

Packaging Overview

- Package Naming: Red Hat software packages follow a standard naming convention. Typically, there are five parts to a package name:

  (1) the package name

  (2) the package version

  (3) the package release (revision or build)

  (4) the Enterprise Linux the package is created for

  (5) the processor architecture the package is built for.

  An installable package name always has the .rpm extension; however this extension is removed from the installed package name.

Packaging Overview

Example:

openssl-1.1.1-8.el8.x86_64.rpm

openssl: package name
1.1.1: version
8: release
el8: stands for Enterprise Linux 8 (not all packages have it)
x86_64: processor architecture the package is created for. You may see “noarch” for platform-independent packages that can be installed on any hardware architecture, or “src” for source code packages.
.rpm: the extension

Package Dependency

- An installable package may require the presence of one or more additional packages in order to be installed successfully. Likewise, a software package component may require the functionality provided by one or more packages to exist in order to operate as expected.

Package Management with rpm(Red Hat Package Manager)

commands:

Displays information file on a package

rpm –qpi [rpmPackage]

Lists out all installed packages

rpm –qpl

Installs specified package

rpm –i[vh] [rpmPackage]

Upgrades an installed with newer version

rpm –U[vh] [rpmPackage]

Package Management with rpm(Red Hat Package Manager)

Erase (Uninstall) an installed package

```
# rpm –e [rpmPackage]
```

S -> Appears if the file size is different
M -> Appears if the (mode) permission or file type is altered
5 -> Appears if MD5 checksum does not match
D -> Appears if the file is a device file and its major or minor number has changed
L -> Appears if the file is a symlink and its path has altered
U -> Appears if the ownership has modified
G -> Appears if the group membership has modified
T -> Appears if timestamp has changed
P -> Appears if capabilities have altered
. -> Appears if no modification is detected

Advanced Package Management Concept and yum

- Package Group: A package group is a collection of correlated packages designed to serve a common purpose. It provides the convenience of querying, installing, and deleting as a single unit rather than dealing with packages individually.

Advanced Package Management Concept and yum

- Application Streams and Modules: Application Streams is a new concept introduced in RHEL 8. It employs a modular approach to organize multiple versions of a software application alongside its dependencies to be available for installation from a single repository. A module can be thought of as a logical set of application packages that includes everything required to install it, including the executables, libraries, documentation, tools, and utilities as well as dependent components. Modularity gives the flexibility to choose the version of software based on need.

Advanced Package Management Concept and yum

- BaseOS Repository: The BaseOS repository includes the core set of RHEL 8 components including the kernel, modules, bootloader, and other foundational software packages. These components lay the foundation to install and run software applications and programs. BaseOS repository components are available in the traditional rpm format.

Advanced Package Management Concept and yum

- The AppStream repository comes standard with core applications, as well as several add-on applications many of them in the traditional rpm format and some in the new modular format. These add-ons include web server software, development languages, database software, etc. and are shipped to support a variety of use cases and deployments.

Advanced Package Management Concept and yum

- yum command:

Searches online repositories for updated packages compared to what is currently installed on the system, upgrades packages

yum update

yum search [package_name]

yum info [package_name]

yum list installed

yum install [package]

yum remove [package]

yum autoremove [package]

yum reinstall [package]

Service Management

- systemd (short for system daemon) is the system initialization and service management mechanism. It has fast-tracked system initialization and state transitioning by introducing features such as parallel processing of startup scripts, improved handling of service dependencies, and on-demand activation of services. Moreover, it supports snapshotting of system states, tracks processes using control groups, and automatically maintains mount points. systemd is the first process with PID 1 that spawns at boot and it is the last process that terminates at shutdown.

Service Management

- Units: Units are systemd objects used for organizing boot and maintenance tasks, such as hardware initialization, socket creation, file system mounts, and service startups. Unit configuration is stored in their respective configuration files, which are auto-generated from other configurations, created dynamically from the system state, produced at runtime, or user-developed. Units are in one of several operational states, including active, inactive, in the process of being activated or deactivated, and failed. Units can be enabled or disabled. An enabled unit can be started to an active state; a disabled unit cannot be started. /usr/lib/systemd/system/ directory is location that all unit files stored. For configuration /etc/systemd/system/ directory is used.
- Targets: Targets are simply logical collections of units. They are a special systemd unit type with the .target file extension. They are also stored in the same directory locations as the other unit configuration files. Targets are used to execute a series of units.

Service Management

- systemctl command:

  # systemctl list-unit-files

  # systemctl list-unit-files

  # systemctl list-dependincies [service]

  # systemctl status [service]

  # systemctl start [service]

  # systemctl stop [service]

  # systemctl enable [service]

  # systemctl disable [service]

  #systemctl is-enabled [service]

  # systemctl is-active [service]

- journalctl command

  # journalctl

Regex (Regular Expression)

- ^ -> starts with

  #^start
- $ -> ends with

  # end$
- Combination of ^ and $

Regex (Regular Expression)

- *-> Any number allowed, but all are optional
  # ab* -> ab, ab[anychar]
- \+-> At least one requir ed;additional are optional # ab\+ -> abc, abcc, abccc
- \?-> One allowed, but it is optional # ab\? -> ab,ab[anychar]
- \{min,max\} -> Min required,max allowed
  # abc\{3,6\} -> abccc,abcccc,abccccc,abcccccc

Regex (Regular Expression)

- | -> Matches either expression it separates

  # abc|abb ->abb,abc
- [...] -> Matches any one character listed

  # ab[bc] ->abb,abc

Regex (Regular Expression)

- \d -> one digit from 0 to 9

  # file_\d -> file_[0-9]
- \w -> word character

  # \w-\w\w\w -> A-b_1
- \s -> whitespace char

  # a\sa\a\s -> a a a
- . -> matches any char
- The upper letters are work complements of the lower letters

- https://www.regular-expressions.info/refquick.html
- https://regex101.com/
- https://www.amazon.com/Regular-Expression-Pocket-Reference-Expressions/dp/0596514271
- https://www.amazon.com/Regular-Expressions-Cookbook-Solutions-Programming/dp/1449319432
- https://www.amazon.com/Introducing-Regular-Expressions-Step-Step/dp/1449392687

grep, egrep, fgrep

- grep command

```
-c      : This prints only a count of the lines that match a pattern
-h      : Display the matched lines, but do not display the filenames.
-i      : Ignores, case for matching
-l      : Displays list of a filenames only.
-n      : Display the matched lines and their line numbers.
-v      : This prints out all the lines that do not matches the pattern
-e exp  : Specifies expression with this option. Can use multiple times.
-f file : Takes patterns from file, one per line.
-E      : Treats pattern as an extended regular expression (ERE)
-w      : Match whole word
-o      : Print only the matched parts of a matching line,
 with each such part on a separate output line.
```

grep, egrep, fgrep

```
# grep -i "UNix" unix.txt
# grep -c  "unix" unix.txt
# grep -l "unix" *
# grep -w "unix" unix.txt
# grep -n "unix" unix.txt
# grep -v "unix" unix.txt
# grep  "^unix" unix.txt
# grep  "$os" unix.txt
```

grep, egrep, fgrep

- egrep and fgrep is similar to grep command. egrep differences is it support more regex expressions than grep. fgrep differences is it can be take input arguments from files.

  # fgrep –f [patternfile] [file]