Software Management and Packaging

coursera.org/learn/linux-for-developers/supplement/448AL/software-management-and-packaging

All Linux distributions group software into packages, which can be defined as a collection of files and subdirectories comprising a product.

If all the software on the system has been installed using the package management utilities, installation, removal, checking integrity and upgrading of software becomes easier and more stable. Of course, there will be other files on the system, such as configuration files and user data, which generally will either be outside the packaging system, or be modified by system administrators from their original content.

There are two main packaging systems in use in Linux systems; **RPM** (used for example in all variations and descendants of Red Hat Enterprise Linux, Fedora and SUSE); and **deb** (used for example in Debian and Ubuntu).

It is important to note that there are always at least two levels of the package management software. The lower level simply installs, updates or removes a package, as in:

```
1

$ sudo rpm -ivh libaio-devel-0.3.109-12.el7.x86_64.rpm
```

which would install the development package associated with the asynchronous I/O library on a Red Hat-derived system. We will shortly show the use of **dpkg** and **apt-get** to do these operations on Debian-derived distributions. However, this will fail if the actual library itself is not installed. Thus, one would have to install them in the proper order, or both at the same time, as in:

```
1

$ sudo rpm -ivh libaio-devel-0.3.109-12.el7.x86_64.rpm libaio-l-0.3.109-
12.el7.x86_64.rpm
```

This can be a process which is both error-prone and frustrating, as you can easily get into dependency hell, where each time you add a package, you find another one missing, and the **rpm** command is not robust enough to always tell you what package it is you need. Furthermore, you have to download the packages first, and supply the full name, including version and architecture, both of which are painful steps.

The higher level utilities handle this automatically. For example, on Red Hat-based systems, the command:

```
1

$ sudo dnf install libaio-devel
```

will resolve any needed dependencies and then download all needed packages from one or more repositories the system has been configured to utilize, and then install them in the proper order. Likewise, the command:

```
1

$ sudo dnf remove libaio
```

will not only remove **libaio** but also **libaio-devel**, since it cannot work without the base package. Obviously, you have to be careful when removing packages, as a cascading effect can happen, but **dnf** will always give you a chance to change your mind before doing it.

Each distribution has such higher level management utilities. For example, **deb**-based systems have **apt-get** and **apt-cache**, while SUSE-based systems have **zypper**. They also have graphical utilities for package management which can insulate you from the command line (older RHEL/CentOS systems use an older program called **dum** in place of **dnf**).

There are a lot of holy wars in the Linux community about which packaging system is the best and there are others besides these two, both older ones and newer cutting edge ones. Often, the criticisms of the various methods are mistakenly applied to the lower level commands, when they should be applied to the higher level ones that deal with dependencies, etc.

Of course, all distributions have graphical system administration utilities for package management, that can do all of this without resort to the low-level commands, but once you know the basic commands, working at the command line tends to be significantly quicker.