6.3.1 Debian Package Manager (dpkg)

Click one of the buttons to take you to that part of the video.

Debian Package Manager (dpkg) 0:04-0:29

In this lesson, we're going to discuss the Debian Package Manager, or dpkg. Now understand that not all Linux distributions use RPM to manage software packages. The distributions that are based on the Debian distribution use the Debian Package Manager instead of RPM. Ubuntu is probably one of the most popular Debian based distributions that uses dpkg in order to manage software instead of RPM.

Debian Packages 0:30-1:48

Understand that Debian packages are similar to RPMs in functionality, but they're totally different in the way that they're actually implemented.

This is important to understand because occasionally I encounter folks who mistakenly think that RPM and Debian software packages are cross platform compatible, meaning you could put, say, an RPM on an Ubuntu system or a Debian package on a Fedora system.

You can't do that. They're not cross platform. You can install RPM packages only on RPM-based systems and Debian packages only on Debian-based systems. However, be aware that there is a utility that you can download called Alien that will convert packages between formats. It doesn't always work perfectly though.

Debian packages use a naming convention that is similar to that that's used by RPM packages. The syntax is shown here. First of all, we have the packagename_version_architecture.deb.

By the way, this is a key clue as to what type of software package you're working with. If it ends in .rpm, it's an RPM package. If it ends in .deb, it's a Debian package. Here's an example of a Debian package, flightgear_3.0.0-5_i386.deb.

---

dpkg 1:49-7:26

In this example, this is the name of the package, this is the version number, and this is the architecture. Now just like RPM packages, Debian packages include dependency information before a given package can be installed on the system.

All of the other packages that it's dependent upon have to be installed first. Now the tool that you use to manage Debian packages is dpkg. And the syntax for using dpkg is shown here. We run dpkg, we use options, we specify an action, and then we specify either the name of the package or the package file name, depending upon whether that package has been installed or not.

If the package has not been installed, you have to use the file name. If the package has been installed or you're accessing it from a repository, then you can just use the name of the package, not the entire file name.

Let's look at how you use dpkg to install a Debian package. Pretty straightforward and the syntax is very similar to that used by RPM. You simply run dpkg -i followed by the name of the package file that you want to install.

Again, because this package isn't currently installed on the system, you do have to specify the full file name, not just the package name. In this example, I'm installing that flightgear package that we looked at earlier. I run dpkg -i, followed by the name of the package.

Now I need to point out right here, notice that before you can install a Debian package, you do have to elevate privileges. On a Debian system we use sudo to temporarily elevate privileges to the root level, and then install the package as shown here.

Now after the package is installed, you can view information about it using the -p option. In this case, you type dpkg -p flightgear. Now just as you can uninstall a package using the RPM utility, you can also uninstall a Debian package using the dpkg command as well.

The option you use is different though. With dpkg we use the -r option for remove, and then the name of the package. For example, if we wanted to uninstall the flightgear package we just installed, we would run dpkg -r flightgear.

Now before we leave this topic, I do need to point out that uninstalling packages with dpkg really is not the best way to do things. It's better to use a package manager like apt-get to uninstall a Debian package, because it's going to calculate dependencies to ensure that the system remains in a stable state before uninstalling a particular package.

For example, let's suppose I ran this command right here, dpkg -r zip. This will uninstall the zip package from my Debian based system.

The problem is that zip is a package that's used by lots of other packages on the system. It's a dependency. Therefore, if I uninstall it with dpkg, all of the other applications on the system that depend upon it are not going to be able to run properly.

Therefore, if you wanted to remove a package, it's actually better to use apt-get instead of dpkg because it will do two things. Number one, it will warn you that there are dependent packages, and then it will prompt you to uninstall all those other packages as well so that you don't end up with broken applications on the system.

If you need to list all of the packages currently installed on the system, you use the dpkg -l command, but be aware that if you do this, the output will be long because there are typically hundreds of packages installed on any given system.

If you're looking for just a specific piece of information, it's best to pipe the output from the dpkg -l command to the input of grep and then tell it what package you're looking for. In this case, we're going to look for the apache2 package.

You can also use dpkg to view package files. For example, if you want to list the files that will be installed by a given package, you can use this command, dpkg -L, followed by the name of the package. In this case, we're looking at apache2.

On the other hand, if you have a file that's already been installed as a part of a package in your file system, in this case we're looking at sshd_config, but you're not sure which package that file came from, which package installed it. You can use the dpkg -S command, and then the dpkg command will try to figure out which package that particular file came from.

Now be aware that this doesn't always work because many times when you install an application on the system, the files that get put in your file system are not copied from the package but are instead created by the package installation process, which the dpkg -S command will not be able to figure out which package performed that action. Only if that file was copied from the package into the file system will it be able to tell you.

Now in addition to the dpkg command, there are several APT tools that you can use to manage packages on a Debian-based system. Now APT stands for Advanced Package Tool, and there are actually lots of different tools within the APT suite. For example, you can use the apt-cache command.

Basically, this is comparable to the RPM -q command in that it's used to query information from the Debian package database, called the package cache.

For example, if we wanted to see if the flightgear package was installed on the system, we would use this command right here: apt-cache pkgnames, and then the name of the package you want to query about, flightgear.

If you want to view information about that package, not just whether it's installed on the system, you would use this command, apt-cache showpkg flightgear. If you needed to view dependency information for that package, you would enter apt-cache depends flightgear.

Summary 7:26-7:40

That's it for this lesson. In this lesson we reviewed how to manage Debian packages. We first reviewed how Debian packages work. Then we talked about using the dpkg command to manage Debian packages. Then we ended this lesson by reviewing how you can use the apt-cache command to get information about Debian packages.
