System Administration for Beginners

Week 6 Notes

March 15, 2010

1 Lecture

Last week we looked at how to install and configure server daemons for UNIX platforms. You'll recall that the process of setting up a server daemon was a process involving the following steps (setting up software for use follows the same general steps):

1. Obtaining the software.
2. Compiling and installing the software.
3. Configuring the software.

It was also mentioned that binaries, or pre-compiled software, could be found for certain UNIX platforms, greatly simplifying the process for people who use that platform. These binaries are usually produced by three groups:

- the people who wrote the software
- enthusiasts would have compiled the software themselves and want to share their work with others
- the people or companies that maintain a certain UNIX platform.

Rarely will you find binaries produced by the people who write software. Compilation of software for various platforms is a very tedious task that usually requires intimate knowledge of the platforms, and programmers are busy people. You are much more likely to find binaries produced by enthusiasts, but you should take their generosity with a grain of salt. There are plenty of malicious people out there – it is very easy to insert a back-door or virus into a binary. If a system administrator were to use an infected binary on a server, the server would be immediately compromised. On the flip side, there are many stupid people on the Internet. It's very easy to improperly configure and compile a piece of software, resulting in an unstable and possibly dangerous binary. For example, you probably wouldn't want to use binaries produced by some 11 year old AOL user on a commercial server.

The best source of binaries are the people or companies who produced your UNIX platform. These people understand your operating system the best and know the best way to configure and compile the software to take advantage of any special features your platform may possess. Furthermore, in the case of commercial UNIX platforms, these people and companies have a financial incentive to produce reliable and secure binaries. Unfortunately not all UNIX platforms are equal – some companies are very diligent in providing binaries and others expect users to compile most software for themselves. The availability of binaries is often a prominent factor when choosing a UNIX platform.

In general, you should first check to see if the company that produces your UNIX platform has binaries available for the software you want to use. If no binaries are available, you should check with the people who produce the software. If they don't provide binaries, your best bet is to compile the software yourself. As a last resort, you can see if any enthusiasts have compiled the software for your platform, but be very careful when deploying those binaries.

1.1 Packages

When you obtain binaries from the people or company that maintains your platform, you'll usually obtain the software in the form of *packages*. Packages are proprietary ways of bundling software so that they can be easily installed, removed, and updated on a specific operating system. Packages can be as low-tech as a compressed tar archive or as complicated as a custom file format that contains automatic configuration software.

To work with packages, you need to use special software provided with your platform. This software understands the package format and usually has features for installing and removing a package. On more advanced platforms, this software often has the ability to also update a package.

The operating system that is installed on your virtual servers is a distribution of Linux called Debian. Debian is a completely free version of Linux that runs on a variety of computer platforms, from small PDAs to large mainframes. You can obtain your own copy of Debian from `debian.org` and install it on your computer at home. Debian uses a special package format called `deb`, and you use a tool called APT (Advanced Package Tool) to work with these packages.

Other distributions of Linux use their own package formats and tools. Besides deb, the most popular format is RPM, and you use a tool of the same name to work with those packages. RPM is the default format for RedHat, Mandrake, and SuSE Linux. However, APT has also been ported to these platforms and has been made compatible with the RPM package format. I personally find APT superior to RPM, and often choose a version of Linux based upon its compatibility with APT, but, as a system administration, you'll be free to choose whatever package format and tool you like best.

I should note that some UNIX platforms take a different approach to packaging. Some platforms do not directly provide binaries, but totally automate the process of compiling and installing software with their package tools. Examples of these platforms are FreeBSD and the Linux distribution Gentoo.

Since your virtual servers use Debian and Debian uses APT for its package management, we'll only be focusing on using APT.

1.2 Package Names

Before you can install a package, you need to know the name of the package you wish to install. Package names are usually based upon the name of the software, but sometimes package maintainers add special tags to package names to differentiate them from similarly named packages. APT maintains a local database of all official Debian packages, and you can query this database using `apt-cache`:

`apt-cache search` ⟨*terms*⟩

`apt-cache search` works in a manner very similar to `apropos`. It'll return a list of packages that match your search terms and a brief description of each package. If you wish to learn more about a package, you can use `apt-cache` to get a package's full description:

`apt-cache show` ⟨*package name*⟩

1.3 Updating APT's Database

Debian continually adds new software to its list of official packages. Before using `apt-cache` to search for packages, you'll want to update the local database of packages in case new packages have been uploaded. You use `apt-get` to update the package database:

```
apt-get update
```

If the package database has been updated, this command will download the latest copy from the Internet. You should also perform this command prior to installing packages, as the package database contains information regarding the version number of each package, and APT can tell you if there are any new versions of the packages you wish to install or have already installed.

1.4 Installing Packages

One of the nice features about APT is that the process of installing packages is automated. When you tell APT to install a package, it will download the package from the Internet, check to see if the package has any *dependencies* (other packages that the original package requires), download the dependencies if necessary, and install all the packages. With other packaging systems, such as RPM, you have to manually download each package and check to see if there are any dependencies, and manually download those packages too.

To install a package using APT, use the following command:

`apt-get install` ⟨*package name*⟩

As with most UNIX commands, you can provide multiple arguments to `apt-get`. For example, if you wish to install multiple packages at the same time, just provide multiple arguments to `apt-get`:

```
apt-get install ⟨package1⟩ ⟨package2⟩
```

In most cases, APT will create a directory in `/usr/share/doc` with the name of the package you've installed. Check the directory for information regarding any special Debian customizations have been applied to the package and any steps you should take before using the package.

Sometimes the documentation will be compressed with `gzip` to conserve disk space. Rather than uncompressing the files and re-compressing them after you're read them, you can use `zcat` and `zless`, which are special versions of `cat` and `less` that understand `gzip`-compressed files.

1.5 Configuring Packages

Another feature that distinguishes APT from other packaging systems is that it integrates configuration with package installation. You'll recall that some software, particularly server daemons, require configuration to work properly. In such a case, APT will conveniently prompt you for more information during the installation of a package. In most cases, APT will also provide you with a reasonable default choice, so if you don't understand the configuration parameter, you can usually safely choose the default option.

Unfortunately, APT can't do all the configuration, so sometimes you'll have to manually tweak a configuration file to fit your needs. For example, when installing Apache, APT will prompt you for the most important settings, but you'll have to edit the Apache configuration file after the installation to specify all the websites you plan on hosting.

1.6 Updating Packages

Yet another feature that makes APT one of the best package management systems is its ability to update packages. With a couple commands, it's possible to update every package that has been installed on your system. First, update your package database:

```
apt-get update
```

Next, use `apt-get` to upgrade all packages that have been updated:

```
apt-get upgrade
```

In this way, APT is similar to Windows Update for Microsoft Windows. However, APT is superior because it updates all software on the system, rather than just software from a specific vendor – Windows Update only updates Microsoft products, whereas APT updates all packages installed on a system.

Software is continually updated to fix bugs and security holes, so it is important to stay on top of all updates. Without a package management software that supports updates, you'd have to download the latest software from a software project's website and go through the tedious process of compiling and installing the software. However, with a package system like APT, the process is automated, and all you have to do is remember to occasionally run the appropriate command.

1.7 Removing Packages

If you've installed software and decided that you didn't like the program, you can uninstall it:

`apt-get remove` ⟨*package*⟩

APT will remove all the software that was installed with the package, but it will retain the configuration files in case you choose to re-install the package at a later date. If you are sure that you'll never re-install a package again, you may specify an additional parameter to APT to instruct it to remove all traces of a package:

`apt-get --purge remove` ⟨*package*⟩