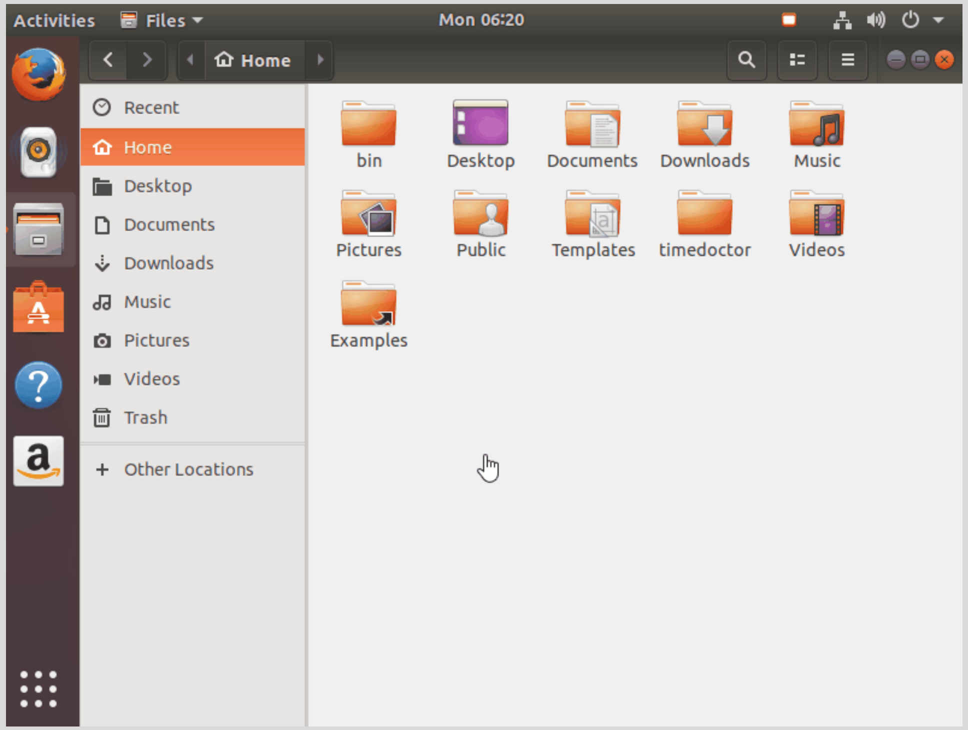
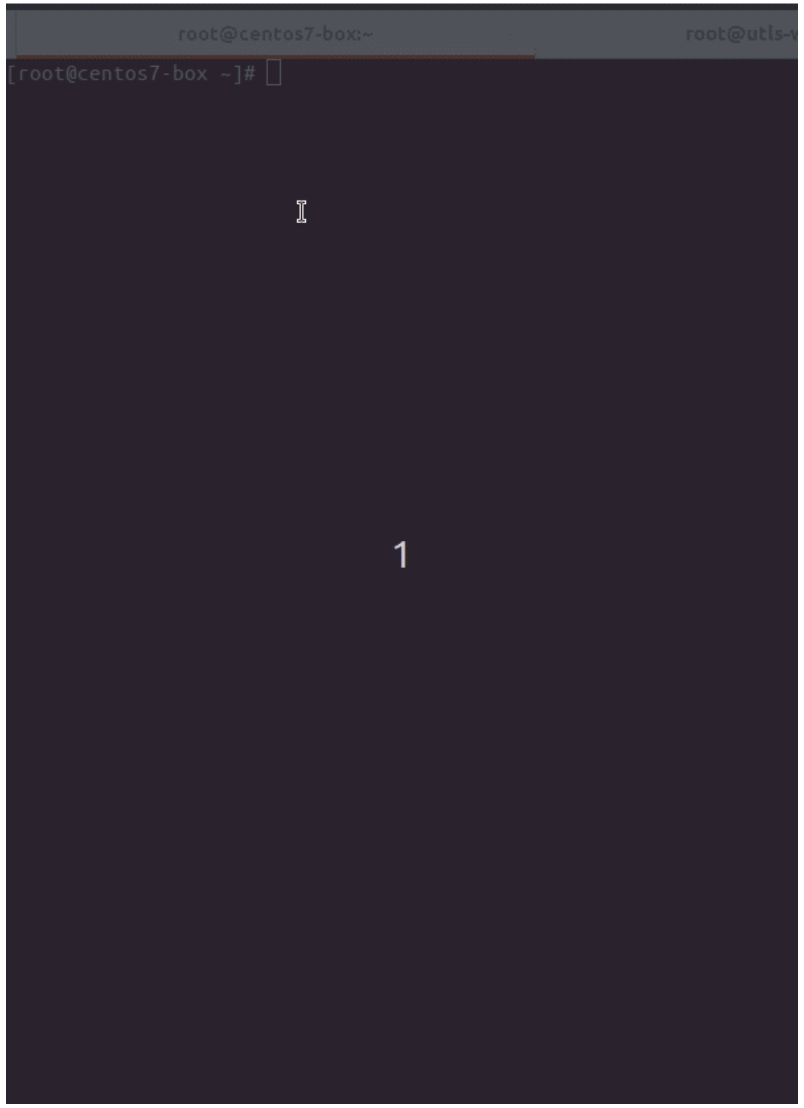
Managing Software

- System administrators deal with software or application management on systems in various ways.
- System administrators like to install the latest and greatest piece of software available out there.
- There are a couple of basic approaches to installing software on a Linux system.
- One approach is to use the ***package manager*** for the distribution.
- A common method for Red Hat–like systems such as Fedora and Red Hat Enterprise Linux (RHEL) is to use the ***Red Hat Package Manager (RPM).***
- The tool of choice for Debian-based systems, such as Ubuntu, Kubuntu, and Debian, is the ***Advanced Packaging Tool (APT)***.
- Another more traditional approach is to compile and install the software by hand using the standard GNU compilation method or the specific software directives.



THE RPM PACKAGE MANAGER

- The Red Hat Package Manager (RPM) allows the easy installation and removal of software packages—typically, precompiled software.
- A package consists of an archive of files and other metadata.
- It is wonderfully easy to use, and several graphical interfaces have been built around it to make it even easier.
- Several Linux distributions (distros) and various third parties use this tool to distribute and package their software.
- An RPM file is a package that contains files needed for the software to function correctly. These files can be configuration files, binaries, and even pre- and postscripts to run while installing the software.
- The RPM tool performs the installation and uninstallation of RPMs.
- The tool also maintains a central database of what RPMs you have installed, where they are installed, when they were installed, and other information about the package.
- In general, software that comes in the form of an RPM is less work to install and maintain than software that needs to be compiled.
- The trade-off is that by using an RPM, you accept the default parameters supplied in the RPM.
- In most cases, these defaults are acceptable. However, if you need to be more intimately aware of what is going on with a piece of software, you may find that by compiling the source yourself, you will learn more about what software components and options exist and how they work together.



```
[root@centos7-box ~]# 
```

There are several great resources for RPM packages, including the following:

- http://rpm.pbone.net
- http://ftp.redhat.com
- http://mirrors.kernel.org
- http://freshrpms.net

The primary functions of the RPM are

❑ Querying, installing, and uninstalling software

❑ Maintaining a database that stores various items of information about the packages

❑ Packaging other software into an RPM form

Following table includes frequently used RPM options, is provided for reference purposes only.

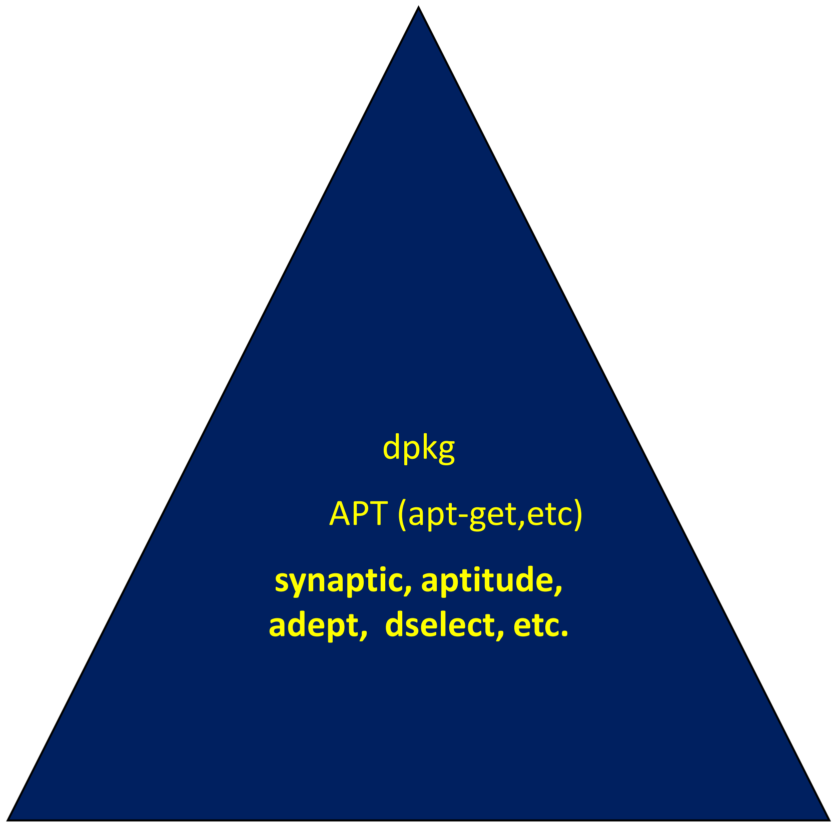
Command-Line Option	Description
--install	This installs a new package.
--upgrade	This upgrades or installs the package currently installed to a newer version.
--erase	Removes or erases an installed package.
--query	This is the option used for querying.
--force	This is the sledgehammer of installation. Typically, you use it when you're knowingly installing an odd or unusual configuration and RPM's safeguards are trying to keep you from doing so. The **--force** option tells RPM to forego any sanity checks and just do it, even if it thinks you're trying to fit a square peg into a round hole. Be careful with this option.
-h	Prints hash marks to indicate progress during an installation. Use with the **-v** option for a pretty display.
--percent	Prints the percentage completed to indicate progress. It is handy if you're running RPM from another program, such as a Perl script, and you want to know the status of the install.
-nodeps	If RPM is complaining about missing dependency files, but you want the installation to happen anyway, passing this option at the command line will cause RPM to not perform any dependency checks.
-q	Queries the RPM system for information.
--test	This option does not perform a real installation; it Just checks to see whether an installation would succeed. If it anticipates problems, it displays what they 'llbe.
-V	Verifies RPMs or files on the system.
-v	Tells RPM to be verbose about its actions.

THE DEBIAN PACKAGE MANAGEMENT SYSTEM

- The Debian Package Management System (DPMS) is the foundation for managing software on Debian and Debian-like systems.
- DPMS provides for easy installation and removal of software packages.
- Debian packages end with the .deb extension.
- At the core of the DPMS is the dpkg (Debian Package) application. dpkg works in the back-end of the system, and several other command-line tools and graphical user interface (GUI) tools have been written to interact with it.

APT

- APT is a highly regarded and sophisticated toolset.
- It is an example of a wrapper tool that interacts directly with dpkg.
- APT is actually a library of programming functions that are used by other middle-ground tools, like **apt-get** and **apt-cache**, to manipulate software on Debian-like systems.
- Several user-land applications have been developed that rely on APT. (*User-land* refers to non-kernel programs and tools.)
- Examples of such applications are synaptic, aptitude, and dselect.
- The user-land tools are generally more user-friendly than their command-line counterparts. APT has also been successfully ported to other operating systems.
- Figure shows what can be described as the DPMS triangle.
- The tool at the apex of the triangle (dpkg) is the most difficult to use and the most powerful, followed by the next easiest to use (APT), and then followed finally by the user-friendly user-land tools.



MANAGING SOFTWARE USING RPM

Querying for All Packages

Use the **rpm** command to list all the packages that are currently installed on your system.

At the shell prompt, type:

```
[root@fedora-serverA ~]# rpm --query --all
```

This will give you a long listing of software installed.

Querying Details for a Specific Package

❖Use **rpm** to see if you indeed have the bash application installed on your system.

```
[root@fedora-serverA ~]# rpm --query bash
bash-3.2-*
```

- The output should be something similar to the one shown.
- It shows that you do indeed have the package called bash installed.
- It also shows the version number appended to the package name.
- Note that the version number of the output on your system might be different; however, the main package name will almost always be the same, i.e., bash is bash is bash in OpenSuSE, Fedora, Mandrake, RHEL, Ubuntu, etc.

❖To lists all the files that come along with the bash package

```
root@fedora-serverA ~]# rpm -ql bash
```

❖To list the configuration files (if any) that come with the bash package, type:

```
[root@serverA ~]# rpm -qc bash
/etc/skel/.bash_logout
/etc/skel/.bash_profile
/etc/skel/.bashrc
```

- RPM packages have a lot of information stored in so-called tags.
- These tags make up the metadata of the package.
- You can query the RPM database for specific information using these tags.

❖For example, to find out the date that the bash package was installed on your system, you can type

```
[root@fedora-serverA ~]# rpm -q --qf "[ %{INSTALLTIME:date} \n]" bash
```

❖Installing with RPM

- This stage of relationships is akin to installing the software package on your system, i.e., moving the software into your system.
- In the following procedures, you will install the application called “lynx” onto your system.
- First, you will need to get a copy of the rpm package for lynx.
- You can get this program from several places (the install CDs/DVD, the Internet, etc.).
- The example that follows uses a copy of the program that came with the DVD used during the installation.
- The CD/DVD needs to be mounted in order to access its content.
- To mount it, insert the DVD into the drive and launch a console.
- You should see an icon for the DVD appear on the desktop after a brief delay.
- The RPM files are stored under the Fedora/RPMS directory under the mount point of your DVD/CD device, e.g., the **/media/dvd/Packages/** directory.

NOTE If you don’t have a Fedora CD or DVD, you can download the RPM we will be using in the next section from http://download.fedora.redhat.com/pub/fedora/linux/releases/9/Everything/i386/os/Packages/lynx-2.8.6-13.fc9.i386.rpm.

```
[root@fedora-serverA ~]# cd /media/dvd/Packages/
```

Let’s step through the process of installing an RPM.

1. Launch a virtual terminal.

2. Assuming your distribution install media disc is mounted at the **/media/dvd** mount point, change to the directory that usually contains the RPM packages on the DVD. Type

3. You can first make sure that the file you want is indeed in that directory. Use the **ls** command to list all the files that start with the letters “lyn” in the directory.

Type

```
[root@fedora-serverA Packages]# ls lyn*
lynx-2.*.rpm
```

4. Now that you have confirmed that the file is there, perform a test install of the package (this will run through the motion of installing the package without actually installing anything on the system). This is useful in making sure that all the needs (dependencies) of a package are met. Type

```
[root@fedora-serverA Packages]# rpm --install --verbose --hash --test lynx-*
Preparing... ################################################ [100%]
```

Everything looks okay., If you get a warning message about the signature, you can safely ignore it for now.

5. Go ahead and perform the actual installation. Type

```
[root@fedora-serverA Packages]# rpm -ivh lynx-*
Preparing.. ########################################### [100%]
     1:lynx ########################################### [100%]
```

6. Run a simple query to confirm that the application is installed on your system.

Type

```
[root@fedora-serverA Packages]# rpm -q lynx
lynx-2.*
```

- The output shows that lynx is now available on the system.
- Lynx is a text-based web browser. You can launch it by simply typing **lynx** at the shell prompt.
- To quit lynx, simply press q.
- You will get a prompt at the lower-right corner of your terminal to confirm that you want to quit lynx.
- Press enter to confirm.

- As you can see, installing packages via RPM can be easy.
- But there are times when installing packages is trickier.
- This is usually due to the issues of *failed dependencies*.
- For example, the lynx package might require the bash package to be already installed on the system before it can be successfully installed.

Let's step through installing a more complex package to see how dependencies are handled with RPM.

Assuming you are still in the Package directory of the DVD media, do the following:

1. Install the package by typing

```
[root@fedora-serverA Packages]# rpm -ivh gcc-4.*
error: Failed dependencies:
glibc-devel >= 2.2.90-12 is needed by gcc-4.3.0-8.i386
```

The preceding output does not look good. The last line tells us that gcc* dependson another package, called glibc-devel*.

2. Fortunately, because we have access to the DVD media that contains most of the packages for this distro in a single directory, we can easily add the additional package to our install list. Type

```
[root@fedora-serverA Packages]# rpm -ivh gcc-4* glibc-devel-2*
error: Failed dependencies:
glibc-headers = 2.8-3 is needed by glibc-devel-2.8-3.i386
```

The output again tells us that the glibc-devel* package depends on another package, called glibc-headers*.

3. Add the newest dependency to the install list. Type

```
[root@fedora-serverA Packages]# rpm -ivh gcc-4* glibc-devel-2*  glibc-headers-2*
error: Failed dependencies:
kernel-headers is needed by glibc-headers-2.8-3.i386
kernel-headers >= 2.2.1 is needed by glibc-headers-2.8-3.i386
```

After all we have given to this relationship, all we get is more complaining. The last requirement is the kernel-headers* package. We need to satisfy this requirement, too.

4. Looks like we are getting close to the end. We add the final required package to the list. Type

```
[root@fedora-serverA Packages]# rpm -ivh gcc-4* glibc-devel-2* | glibc-headers-2* kernel-headers-*
Preparing... ########################################### [100%]
1:kernel-headers ########################################### [ 25%]
2:glibc-headers ########################################### [ 50%]
3:glibc-devel ########################################### [ 75%]
4:gcc ########################################### [100%]
```

A popular option used in installing packages via RPM is the **-U** (for Upgrade) option.

It is especially useful when you want to install a newer version of a package that already exists. It will simply upgrade the already installed package to the newer version.

❖Uninstalling Software with RPM

- Cleaning up after itself is one of the areas in which RPM truly excels.
- And this is one of its key selling points as a software manager in Linux systems.
- Because a database of various pieces of information is stored and maintained along with each installed package, it is easy for RPM to refer back to its database to collect information about what was installed and where.
- Removing software with RPM is quite easy and can be done in a single step.

For example, to remove the lynx package that we installed earlier, we simply need to use the **-e** option, like so:

```
[root@fedora-serverA ~ ]# rpm -e lynx
```

- This command will usually not give you any feedback if everything went well.
- To get a more verbose output for the uninstallation process, add the **-v** option to the command.

A handy feature of RPM is that it will also protect you from removing packages that are needed by other packages. For example, if we try to remove the kernel-headers package (recall that the gcc package depended on it), we'd see the following:

```
[root@fedora-serverA ~]# rpm -e kernel-headers
error: Failed dependencies:
kernel-headers is needed by (installed) glibc-headers-2.8-3.i386
kernel-headers >= 2.2.1 is needed by (installed) glibc-headers-2.8-3.i386
```

❖Other Things You Can Do with RPM

In addition to basic installation and uninstallation of packages with RPM, there are numerous other things you can do with it.

❖Verifying Packages

A useful option with the RPM tool is the ability to verify a package.

What happens is that RPM looks at the package information in its database, which is assumed to be good. It then compares that information with the binaries and files that are on your system.

In today's Internet world, where being hacked is a real possibility, this kind of test should tell you instantly if anyone has done something to your system. For example, to verify that the bash package is as it should be, type

```
[root@fedora-serverA ~]# rpm -V bash
```

The absence of any output is a good sign. You can also verify specific files on the file system that a particular package installed.

For example, to verify that the **/bin/ls** command is valid, you would type

```
[root@fedora-serverA ~ ]# rpm -Vf /bin/ls
```

Again, the lack of output is a good thing.

❖ Yum

- Yum is one of the newer methods of software management on Linux systems.
- It is basically a wrapper program for RPM, with great enhancements.
- It has been around for a while, but it has become more widely used and more prominent because major Linux vendors decided to concentrate on their (more profitable) commercial product offerings.
- Yum has changed and enhanced the traditional approach to package management on RPM-based systems.
- Popular large sites that serve as repositories for open source software have had to retool slightly to accommodate "Yumified" repositories. According to the Yum project's web page:

"Yum is an automatic updater and package installer/remover for RPM systems. It automatically computes dependencies and figures out what things should occur to install packages. It makes it easier to maintain groups of machines without having to manually update each one using RPM."

You mostly need a single configuration file (**/etc/yum.conf**). Other configuration files may be stored under the **/etc/yum.repos.d/** directory that points to the Yum-enabled (Yumified) software repository. Fortunately, several Linux distributions now ship with Yum already installed and preconfigured.

Fedora is one of these distros. To use Yum on a Fedora system (or any other Red Hat–like distro)—to install a package called gcc, for example—at the command line, you would type

```
[root@fedora-serverA ~]# yum install gcc
```

Yum will automatically take care of any dependencies that the package might need and install the package for you.

- Yum also has extensive search capabilities that will help you find a package, even if you don't know its correct name. All you need to know is part of the name.

For example, if you wanted to search for all packages that have the word "headers" in the name, you can try a Yum option like this:

```
[root@fedora-serverA ~]# yum search headers
```

This will return a long list of matches. You can then look through the list and pick the package you want.

❖SOFTWARE MANAGEMENT IN UBUNTU

- Software management in the Debian-like distros such as Ubuntu is done using DPMS and all the attendant applications built around it, such as APT and dpkg.
- In this section we will look at how to perform basic software management tasks on Debian-like distros.

Querying for Information

On your Ubuntu server, the equivalent command to list all currently installed software is

```
yyang@ubuntu-server:~$ dpkg –l
```

The command to get basic information about an installed package is

```
yyang@ubuntu-server:~$ dpkg -l bash
```

The command to get more detailed information about the bash package is

```
yyang@ubuntu-server:~$ dpkg --print-avail bash
```

To view the list of files that comes with the bash package, type

```
yyang@ubuntu-server:~$ dpkg-query -L bash
```

The querying capabilities of dpkg are extensive. You can use DPMS to query for specific information about a package. For example, to find out the size of the installed bash package, you can type

```
yyang@ubuntu-server:~$ dpkg-query -W --showformat='${Package} ${Installed-Size} bash
```

❖Installing Software in Ubuntu

- There are several ways to get software installed on Ubuntu systems.
- You can use dpkg to directly install a .deb file, or you may choose to use **apt-get** to install any software available in the Ubuntu repositories on the Internet or locally (CD/DVD ROM, file system, etc).
- To use dpkg to install a .deb package named lynx_2.8.6-2ubuntu2_i386.deb , type

```
yyang@ubuntu-serverA:~$ sudo dpkg --install lynx_2.8.6-2ubuntu2_i386.deb
```

- Using **apt-get** to install software is a little easier, because APT will usually take care of any dependency issues for you.
- The other advantage to using APT to install software is that you only need to know a part of the name of the software; you don’t need to know the exact version number.
- You also don’t need to manually download the software before installing.
- To use **apt-get** to install a package called lynx, type

```
yyang@ubuntu-server:~$ sudo apt-get install lynx
```

❖Removing Software in Ubuntu

Uninstalling software in Ubuntu using dpkg is as easy as typing

```
yyang@ubuntu-server:~$ sudo dpkg --remove lynx
```

You can also use **apt-get** to remove software by using the **remove** option. To remove the lynx package using **apt-get**, type

```
yyang@ubuntu-server:~$ sudo apt-get remove lynx
```

APT makes it easy to completely remove software and any attendant configuration file(s) from a system.

This allows you to truly start from scratch by getting rid of any customized configuration files.

Assuming we completely want to remove the lynx application from the system, we would type

```
yyang@ubuntu-server:~$ sudo apt-get --purge remove lynx
```

❖ GUI RPM Package Managers

For those who like a good GUI tool to help simplify their lives, several package managers with UI front-ends are available.

Doing all the dirty work behind these pretty GUI front-ends is RPM.

The GUI tools allow you to do quite a few things without forcing you to remember command-line parameters.

Some of the more popular ones with each distribution or desktop environment are listed in the sections that follow.

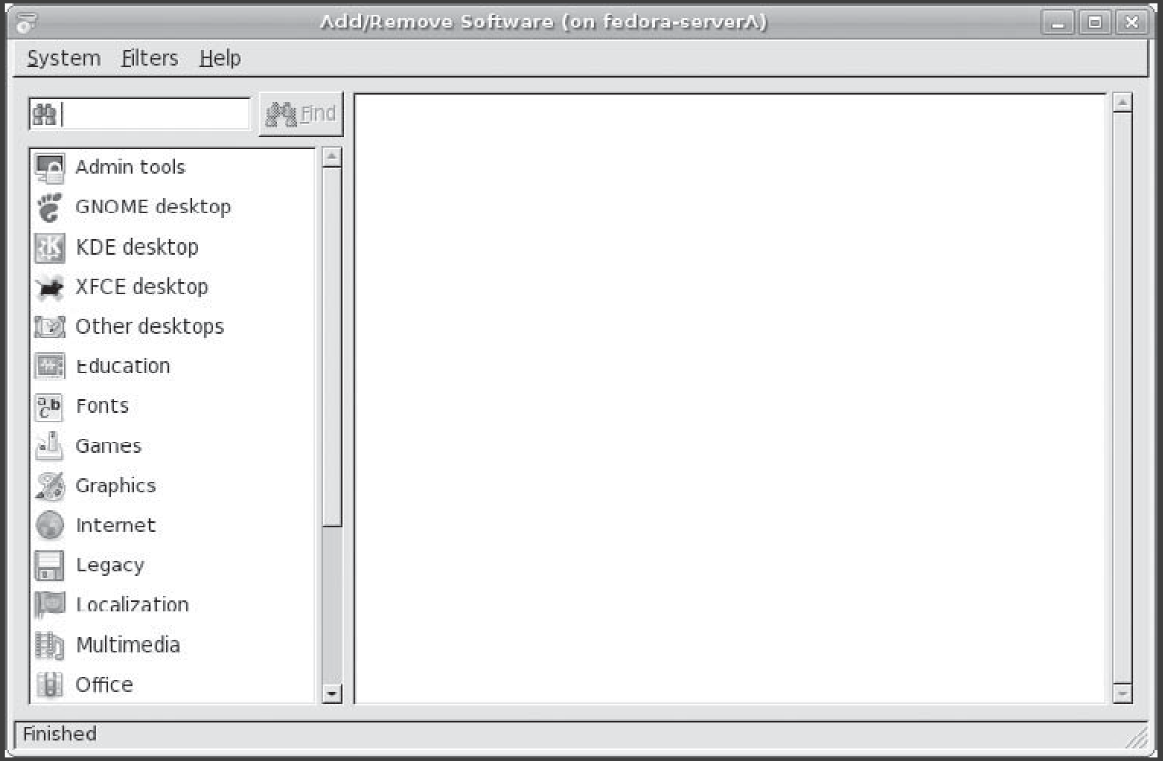
❖ Fedora

You can launch the GUI package management tool (see Figure) in Fedora by selecting

System menu | Administration | Add/Remove Software.

You can also launch the Fedora package manager from the command line, simply by typing

```
[root@fedora-serverA ~]# gpk-application
```



❖ OpenSuSE and SLE

In OpenSuSe and SuSE Linux Enterprise (SLE), most of the syste
administration is done via a tool called YaST, which stands for Y
Another Setup Tool.

YaST is made up of different modules. For adding and removing
packages graphically on the system, the relevant module is calle
sw_single.

So to launch this module from the command line of a system running the SuSE distribution, you would type

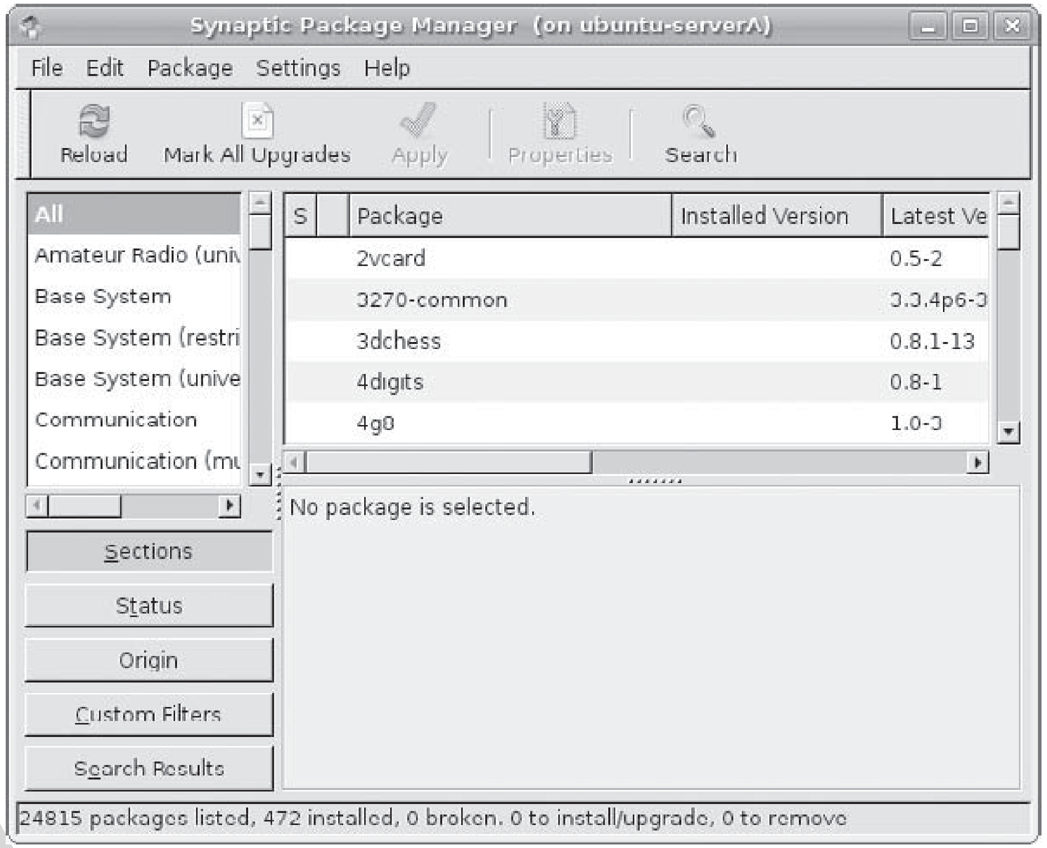
```
suse-serverA:~ # yast2 sw_single
```

❖ Ubuntu

Several GUI software management tools are available on Ubuntu systems. For desktopclass systems, GUI tools are installed by default. Some of the more popular GUI tools in Ubuntu are synaptic and adept.

Ubuntu also has a couple of tools that are not exactly GUI, but offer a similar ease of use as their fat GUI counterparts.

These tools are console-based or text-based and menu-driven. Examples of such tools are aptitude and dselect.



S		Package	Installed Version	Latest Ve
		2vcard		0.5-2
		3270-common		3.3.4p6-3
		3dchess		0.8.1-13
		4digits		0.8-1
		4g8		1.0-3

No package is selected.

24815 packages listed, 472 installed, 0 broken. 0 to install/upgrade, 0 to remove