

By Jack Wallen

Installing software in Linux is nothing like it used to be. If you follow this little guide, your Linux life will be made simpler and safer.

1 Installing from source when your system is primarily an .rpm or .deb system

Many new Linux users don't understand that both rpm and apt (or dpkg) keep track of everything installed on the system. However, those systems (rpm, apt, and dpkg) can keep track only of packages they install. So when you find that obscure package that comes only in source and you compile it yourself, your package management system will not know what to do with it. Instead, create either an .rpm or .deb file from the source and install the package with the package management system so that system will be aware of everything you have installed.

2 Neglecting the many graphical front-end package management applications

Most people don't even realize that there are graphical front ends that take a lot of the guesswork out of installing packages in Linux. For yum (the command-line package management system for rpm), you can use Yumex for yum (installed with `yum install yumex`); you can use Synaptic or Adept for apt (installed with `apt-get install synaptic` or `apt-get install adept`).

3 Forgetting to update the list of available packages

When using apt-get or yum, make sure you're updating the list of available packages. Otherwise, your system will not remain updated with the latest releases of installed packages. To update with apt-get, you issue the command `apt-get update`. To update with yum, issue `yum check-update`.

4 Not adding repositories for yum or apt-get

Both yum and apt-get use a listing of repositories that tell them where to locate available packages. But the default repositories (often called "repos") do not include every Linux package known to Linuxkind. So if you run the command to install an application, and yum (or apt-get) can't find the package, most likely you'll have to add a repo to your sources listing.

For yum, the sources are in `/etc/yum.conf`. For apt-get, they are placed in `/etc/apt/sources.list`. Once you have added a new repo, make sure you run the update so either apt or rpm is made aware of the new source.

5 Not taking advantage of installing from a browser

Just as with Windows, when your system sees you are attempting to download an installable application, you'll be asked whether you would like the package management system to attempt to install the file or just save it to disk. In both instances, you will be asked for the root password (so you must have access to said password for this to even work). One thing I've always like about this method (be it in a yum-based or dpkg-based system) is that it has almost always been good about locating and adding dependencies.

Naturally, this method works only when you are downloading a file that's applicable to your system. If you attempt to download an rpm file on a Debian-based system, you won't have the option of installing the file.

You can take this one step further and select the Always Do This... check box in the Firefox popup so that every time you download a file associated with your package management system, it will automatically prompt you for your root password and continue to install the package. This streamlines the process quite a bit.

6 Forgetting the command line

Let's say you've installed a headless server using Ubuntu or Debian (a common setup for Linux servers) and haven't installed any of the graphical interfaces or desktops. To do any maintenance, you have to log in via ssh (because no admin would log in via telnet) and are limited to the command line only. Even so, your ability to keep your system updated or install new applications is not limited. You can still use yum or apt-get to manage your packages.

With a Debian-based system, you have another option: Aptitude. From the command line, issue the command *aptitude* and you will be greeted with a nice curses-based interface for apt. This system is easy to use and gives you an outstanding option for maintaining a gui-less server without losing functionality. Aptitude lists Security Updates, Upgradeable Packages, New Packages, Not Installed Packages, Obsolete Packages, Virtual Packages, and Tasks. As you scroll through the list, you will not only get the installed vs. the new package release numbers but also a description of the package. After using Aptitude, you will quickly see how simple updating Linux packages can be, even from the command line.

7 Blindly unpacking tar files

I can't tell you how many times I have downloaded a source package and without thinking, untarred the package not knowing its contents. Most times this works out fine. But there are a few times when the package creator/maintainer has failed to mention that the entire contents of the package are not housed in a parent directory. So instead of having a newly created directory housing the contents of the tar file (which can contain hundreds of files/directories), those files are blown up into the directory you unpacked them into.

To avoid this, I always create a temporary directory and move the tar file into it. Then, when I unpack the tar file, it doesn't matter if the contents are contained within their own directory or not. Using this method will save you a LOT of cleanup in those cases where the creator didn't pack everything in its own neat directory.

8 Deleting those make files

When you're installing from source, you'll probably run *make clean* to get rid of all of those unneeded source files. But if you get rid of the Makefile, uninstalling will be a hassle. If you keep it, you can usually uninstall the program simply by issuing *make uninstall* from the directory housing the Makefile.

A word of warning: Don't dump all your Makefiles into one directory. First rename them so you know which application they belong to. When you want to uninstall the application, move the Makefile to another directory, rename it to its original name, and then run the *uninstall* command. Once you've uninstalled the application, you can delete the Makefile.

9 Installing for the wrong architecture

You might notice that many rpm files will have an i386, i586, i686, PPC, 64, etc. There is a reason for this. Unless the rpm file has *noarch* included in the filename, that rpm file was created for a specific architecture. And when those files were created for that architecture, they were optimized for it, so they'll run better. Does that mean you can't install an i586 on a standard 386 machine? Of course not. But it will not run as efficiently as it will on the indicated architecture. Now, you can't install a PPC rpm on an x86 architecture. The PPC architecture is for the Motorola chipset. Nor can you install the 64 bit on a 32 bit. You can, however, install the 32 bit on a 64 bit (as in the case when you want to get Firefox running with Flash on a 64-bit machine).

10 Failing to address problems with kernel updates

It used to be that updating kernels was a task left to the silverback geeks. No more. With the new package management systems, anyone can update a kernel. But there are some gotchas you should know about. One issue is that of space. With every update of a kernel, your old kernel is retained. If you continually update kernels, your system storage can quickly fill up. It's always a good idea to check to see what older kernels you can get rid of. If you're using rpm, issue the command `rpm -qa | grep kernel` to see what you have installed. You can remove all but the last two installed. It's always best to keep two in case the one you are running gets fubar'd.

Another gotcha involves NVIDIA drivers. If you use the livna repositories, you will find yourself locked into the livna kernel releases as well. This isn't always a good idea. Instead, I would do this in two parts: Update your kernel and then download and install the NVIDIA drive associated with your kernel. This will require you to search for the proper rpm file for the NVIDIA driver, but it will keep you from having to use the livna kernel. I was once locked into this system and found myself suffering from interesting kernel/video issues isolated to the livna files. Avoid this. Of course if you are using a Ubuntu system you can avoid the NVIDIA trap altogether by using [Envy](http://albertomilone.com/nvidia_scripts1.html) (http://albertomilone.com/nvidia_scripts1.html). This handy tool will allow you to install the best NVIDIA driver without having to mess up your favorite kernel.

And although this is a no brainer, make sure you reboot after a kernel upgrade. It's the one time you will HAVE to reboot your Linux machine. Although your machine will continue to work just fine, it will be working with the older kernel and not taking advantage of the new feature or security enhancement (or whatever the newer kernel has to offer).

Additional resources

- TechRepublic's [Downloads RSS Feed](#) [XML](#)
- Sign up for the [Downloads at TechRepublic](#) newsletter
- Sign up for our [Linux NetNote](#)
- Check out all of TechRepublic's [free newsletters](#)
- [10 things to consider when choosing a Linux distribution](#)
- [How do I... Install plugins for Firefox in Linux?](#)

Version history

Version: 1.0

Published: April 14, 2008