



## Package Management Basics: apt, yum, dnf, pkg

Posted January 4, 2016 365.3k

LINUX BASICS

GETTING STARTED

SYSTEM TOOLS

LINUX COMMANDS

By [Brennen Bearnes](#)

[Become an author](#)

### Introduction

Most modern Unix-like operating systems offer a centralized mechanism for finding and installing software. Software is usually distributed in the form of **packages**, kept in **repositories**. Working with packages is known as **package management**. Packages provide the basic components of an operating system, along with shared libraries, applications, services, and documentation.

A package management system does much more than one-time installation of software. It also provides tools for upgrading already-installed packages. Package repositories help to ensure that code has been vetted for use on your system, and that the installed versions of software have been approved by developers and package maintainers.

When configuring servers or development environments, it's often necessary look beyond official repositories. Packages in the stable release of a distribution may be out of date, especially where

new or rapidly-changing software is concerned. Nevertheless, package management is a vital skill for system administrators and developers, and the wealth of packaged software for major distributions is a tremendous resource.

This guide is intended as a quick reference for the fundamentals of finding, installing, and upgrading packages on a variety of distributions, and should help you translate that knowledge between systems.

---

## Package Management Systems: A Brief Overview

Most package systems are built around collections of package files. A package file is usually an archive which contains compiled binaries and other resources making up the software, along with installation scripts. Packages also contain valuable metadata, including their **dependencies**, a list of other packages required to install and run them.

While their functionality and benefits are broadly similar, packaging formats and tools vary by platform:

Operating System	Format	Tool(s)
Debian	.deb	apt , apt-cache , apt-get , dpkg
Ubuntu	.deb	apt , apt-cache , apt-get , dpkg
CentOS	.rpm	yum
Fedora	.rpm	dnf
FreeBSD	Ports, .txz	make , pkg

In Debian and systems based on it, like Ubuntu, Linux Mint, and Raspbian, the package format is the `.deb` file. APT, the Advanced Packaging Tool, provides commands used for most common operations: Searching repositories, installing collections of packages and their dependencies, and managing upgrades. APT commands operate as a front-end to the lower-level `dpkg` utility, which handles the installation of individual `.deb` files on the local system, and is sometimes invoked directly.

Recent releases of most Debian-derived distributions include the `apt` command, which offers a concise and unified interface to common operations that have traditionally been handled by the

more-specific `apt-get` and `apt-cache`. Its use is optional, but may simplify some tasks.

CentOS, Fedora, and other members of the Red Hat family use RPM files. In CentOS, `yum` is used to interact with both individual package files and repositories.

In recent versions of Fedora, `yum` has been supplanted by `dnf`, a modernized fork which retains most of `yum`'s interface.

FreeBSD's binary package system is administered with the `pkg` command. FreeBSD also offers the Ports Collection, a local directory structure and tools which allow the user to fetch, compile, and install packages directly from source using Makefiles. It's usually much more convenient to use `pkg`, but occasionally a pre-compiled package is unavailable, or you may need to change compile-time options.

---

## Update Package Lists

Most systems keep a local database of the packages available from remote repositories. It's best to update this database before installing or upgrading packages. As a partial exception to this pattern, `yum` and `dnf` will check for updates before performing some operations, but you can ask them at any time whether updates are available.

System	Command
Debian / Ubuntu	<code>sudo apt-get update</code>
	<code>sudo apt update</code>
CentOS	<code>yum check-update</code>
Fedora	<code>dnf check-update</code>
FreeBSD Packages	<code>sudo pkg update</code>
FreeBSD Ports	<code>sudo portsnap fetch update</code>

---

## Upgrade Installed Packages

Making sure that all of the installed software on a machine stays up to date would be an enormous undertaking without a package system. You would have to track upstream changes and

security alerts for hundreds of different packages. While a package manager doesn't solve every problem you'll encounter when upgrading software, it does enable you to maintain most system components with a few commands.

On FreeBSD, upgrading installed ports can introduce breaking changes or require manual configuration steps. It's best to read `/usr/ports/UPDATING` before upgrading with `portmaster`.

System	Command	Notes
Debian / Ubuntu	<code>sudo apt-get upgrade</code>	Only upgrades installed packages, where possible.
	<code>sudo apt-get dist-upgrade</code>	May add or remove packages to satisfy new dependencies.
	<code>sudo apt upgrade</code>	Like <code>apt-get upgrade</code> .
	<code>sudo apt full-upgrade</code>	Like <code>apt-get dist-upgrade</code> .
CentOS	<code>sudo yum update</code>	
Fedora	<code>sudo dnf upgrade</code>	
FreeBSD Packages	<code>sudo pkg upgrade</code>	
FreeBSD Ports	<code>less /usr/ports/UPDATING</code>	Uses <code>less</code> to view update notes for ports (use arrow keys to scroll, press <b>q</b> to quit).
	<code>cd /usr/ports/ports-mgmt/portmaster &amp;&amp; sudo make install &amp;&amp; sudo portmaster -a</code>	Installs <code>portmaster</code> and uses it to update installed ports.

---

## Find a Package

Most distributions offer a graphical or menu-driven front end to package collections. These can be a good way to browse by category and discover new software. Often, however, the quickest and most effective way to locate a package is to search with command-line tools.

System	Command	Notes
Debian / Ubuntu	<code>apt-cache search <span style="color: red;">search_string</span></code>	

System	Command	Notes
	apt search <code>search_string</code>	
CentOS	yum search <code>search_string</code>	
	yum search all <code>search_string</code>	Searches all fields, including description.
Fedora	dnf search <code>search_string</code>	
	dnf search all <code>search_string</code>	Searches all fields, including description.
FreeBSD Packages	pkg search <code>search_string</code>	Searches by name.
	pkg search -f <code>search_string</code>	Searches by name, returning full descriptions.
	pkg search -D <code>search_string</code>	Searches description.
FreeBSD Ports	cd /usr/ports && make search name= <code>package</code>	Searches by name.
	cd /usr/ports && make search key= <code>search_string</code>	Searches comments, descriptions, and dependencies.

## View Info About a Specific Package

When deciding what to install, it's often helpful to read detailed descriptions of packages. Along with human-readable text, these often include metadata like version numbers and a list of the package's dependencies.

System	Command	Notes
Debian / Ubuntu	apt-cache show <code>package</code>	Shows locally-cached info about a package.
	apt show <code>package</code>	
	dpkg -s <code>package</code>	Shows the current installed status of a package.
CentOS	yum info <code>package</code>	

System	Command	Notes
	<code>yum deplist <b>package</b></code>	Lists dependencies for a package.
Fedora	<code>dnf info <b>package</b></code>	
	<code>dnf repoquery --requires <b>package</b></code>	Lists dependencies for a package.
FreeBSD Packages	<code>pkg info <b>package</b></code>	Shows info for an installed package.
FreeBSD Ports	<code>cd /usr/ports/<b>category/port</b> &amp;&amp; cat pkg-descr</code>	

## Install a Package from Repositories

Once you know the name of a package, you can usually install it and its dependencies with a single command. In general, you can supply multiple packages to install simply by listing them all.

System	Command	Notes
Debian / Ubuntu	<code>sudo apt-get install <b>package</b></code>	
	<code>sudo apt-get install <b>package1 package2 ...</b></code>	Installs all listed packages.
	<code>sudo apt-get install -y <b>package</b></code>	Assumes "yes" where apt would usually prompt to continue.
	<code>sudo apt install <b>package</b></code>	Displays a colored progress bar.
CentOS	<code>sudo yum install <b>package</b></code>	
	<code>sudo yum install <b>package1 package2 ...</b></code>	Installs all listed packages.
	<code>sudo yum install -y <b>package</b></code>	Assumes "yes" where yum would usually prompt to continue.
Fedora	<code>sudo dnf install <b>package</b></code>	
	<code>sudo dnf install <b>package1 package2 ...</b></code>	Installs all listed packages.

System	Command	Notes
	<code>sudo dnf install -y package</code>	Assumes "yes" where dnf would usually prompt to continue.
FreeBSD Packages	<code>sudo pkg install package</code>	
	<code>sudo pkg install package1 package2 ...</code>	Installs all listed packages.
FreeBSD Ports	<code>cd /usr/ports/category/port &amp;&amp; sudo make install</code>	Builds and installs a port from source.

## Install a Package from the Local Filesystem

Sometimes, even though software isn't officially packaged for a given operating system, a developer or vendor will offer package files for download. You can usually retrieve these with your web browser, or via `curl` on the command line. Once a package is on the target system, it can often be installed with a single command.

On Debian-derived systems, `dpkg` handles individual package files. If a package has unmet dependencies, `gdebi` can often be used to retrieve them from official repositories.

On CentOS and Fedora systems, `yum` and `dnf` are used to install individual files, and will also handle needed dependencies.

System	Command	Notes
Debian / Ubuntu	<code>sudo dpkg -i package.deb</code>	
	<code>sudo apt-get install -y gdebi &amp;&amp; sudo gdebi package.deb</code>	Installs and uses <code>gdebi</code> to install <code>package.deb</code> and retrieve any missing dependencies.
CentOS	<code>sudo yum install package.rpm</code>	
Fedora	<code>sudo dnf install package.rpm</code>	
FreeBSD Packages	<code>sudo pkg add package.txz</code>	
	<code>sudo pkg add -f package.txz</code>	Installs package even if already installed.

---

## Remove One or More Installed Packages

Since a package manager knows what files are provided by a given package, it can usually remove them cleanly from a system if the software is no longer needed.

System	Command	Notes
Debian / Ubuntu	<code>sudo apt-get remove <b>package</b></code>	
	<code>sudo apt remove <b>package</b></code>	
	<code>sudo apt-get autoremove</code>	Removes unneeded packages.
CentOS	<code>sudo yum remove <b>package</b></code>	
Fedora	<code>sudo dnf erase <b>package</b></code>	
FreeBSD Packages	<code>sudo pkg delete <b>package</b></code>	
	<code>sudo pkg autoremove</code>	Removes unneeded packages.
FreeBSD Ports	<code>sudo pkg delete <b>package</b></code>	
	<code>cd /usr/ports/<b>path_to_port</b> &amp;&amp; make deinstall</code>	De-installs an installed port.

---

## The apt Command

Administrators of Debian-family distributions are generally familiar with `apt-get` and `apt-cache`. Less widely known is the simplified `apt` interface, designed specifically for interactive use.

Traditional Command	apt Equivalent
<code>apt-get update</code>	<code>apt update</code>
<code>apt-get dist-upgrade</code>	<code>apt full-upgrade</code>
<code>apt-cache search <b>string</b></code>	<code>apt search <b>string</b></code>
<code>apt-get install <b>package</b></code>	<code>apt install <b>package</b></code>



Traditional Command	apt Equivalent
apt-get remove <b>package</b>	apt remove <b>package</b>
apt-get purge <b>package</b>	apt purge <b>package</b>

While `apt` is often a quicker shorthand for a given operation, it's not intended as a complete replacement for the traditional tools, and its interface may change between versions to improve usability. If you are using package management commands inside a script or a shell pipeline, it's a good idea to stick with `apt-get` and `apt-cache`.

## Get Help

In addition to web-based documentation, keep in mind that Unix manual pages (usually referred to as **man pages**) are available for most commands from the shell. To read a page, use `man`:

`man page`

In `man`, you can navigate with the arrow keys. Press `/` to search for text within the page, and `q` to quit.

System	Command	Notes
Debian / Ubuntu	<code>man apt-get</code>	Updating the local package database and working with packages.
	<code>man apt-cache</code>	Querying the local package database.
	<code>man dpkg</code>	Working with individual package files and querying installed packages.
	<code>man apt</code>	Working with a more concise, user-friendly interface to most basic operations.
CentOS	<code>man yum</code>	
Fedora	<code>man dnf</code>	
FreeBSD Packages	<code>man pkg</code>	Working with pre-compiled binary packages.

System	Command	Notes
FreeBSD Ports	<code>man ports</code>	Working with the Ports Collection.

.....

## Conclusion and Further Reading

This guide provides an overview of basic operations that can be cross-referenced between systems, but only scratches the surface of a complex topic. For greater detail on a given system, you can consult the following resources:

- [This guide](#) covers Ubuntu and Debian package management in detail.
- There's an [official CentOS guide](#) to managing software with `yum`.
- There's a [Fedora wiki page](#) about `dnf`, and an [official manual](#) for `dnf` itself.
- [This guide](#) covers FreeBSD package management using `pkg`.
- The [FreeBSD Handbook](#) contains a [section](#) on using the Ports Collection.