You may know some software like Firefox, GIMP, etc, but what they really are is just lots and lots of files that have been compiled into one. The people (or sometimes a single person) that write this software are known as upstream providers, they compile their code and write up how to get it installed.

These upstream providers work on getting new software out and update existing software. When they are ready to release it to the world, they send their package to package maintainers who handle getting this piece of software in the hands of the users. These package maintainers review, manage, and distribute this software in the form of packages. So instead of downloading packages/software one by one from random sites (like on Windows), there is something better -- called package repositories in Debian.

Repositories are just a central storage location for packages. Many online repositories hold lots of packages. Your machine doesn't know where to look for these repositories unless you explicitly tell it where to look. Debian already comes with pre-approved sources to get packages from and this is how it installs all the base packages you see on your system (if a user did a net-install). On a Debian system, this sources file is the "/etc/apt/sources.list" file. Your machine will know to look there and check for any source repositories you added.

Apt

*apt* provides a high-level command line interface for debian's package management system, with all the necessary tools in one place. *apt* aims to provide a simple, efficient way of handling packages for end users.

**Update your package lists**

To use apt, it is important to retrieve a fresh copy of the package lists from the configured sources:

# apt update

**Upgrading your system**

You should upgrade you system everyday, this gives you security and bug fixes:

# apt upgrade

Sometimes it is helpful to use "full-upgrade". It performs the function of the upgrade but will remove currently installed packages if that is essential to upgrade the system as a whole. This command is usually issued while doing system upgrades.

# apt full-upgrade

Before upgrading your Debian system to the next release, drop into a "[tty](https://wiki.debian.org/getty)" console by pressing CTRL+ALT+F3/F4/F5 etc, as dpkg restarts some login managers that will forcefully log you out while upgrading, resulting in a broken/partial upgrade

**Installing and removing packages**

To install a package, in this example Firefox:

# apt install firefox

To remove a package:

# apt remove firefox

You can also install a .deb file with:

# apt install </path/to/deb/file/debfile.deb>

**Note**

*apt* performs the requested action on one or more packages specified via regex, glob or exact match. The requested action can be overridden for specific packages by appending a plus (+) to the package name to install this package or a minus (-) to remove it.

A specific version of a package can be selected for installation by following the package name with an equals (=) and the version of the package to select. Alternatively, the version from a specific release can be selected by following the package name with a forward slash (/) and code name (stretch, buster, sid...) or suite name (stable, testing, unstable). This will also select versions from this release for dependencies of this package if needed to satisfy the request. Removing a package removes all packaged data, but leaves usually small (modified) user configuration files behind, in case the remove was an accident. Just issuing an installation request for the accidentally removed package will restore its function as before in that case. On the other hand, you can get rid of these leftovers by calling purge even on already removed packages. Note that this does not affect any data or configuration stored in the users' home directory.

**Finding packages**

Search for a package:

$ apt search <searchterm>

For example, to find packages with the term 'cats' in the package name or description:

$ apt search cats

To see detailed information about a package:

$ apt info chameleon-cursor-theme

**Clean up**

After you remove a package, you should remove any other packages that were automatically installed to satisfy dependencies and are now no longer needed:

# apt autoremove

It can also be issued while removing a package like:

# apt remove clang --auto-remove

You should check this does not include applications you have grown to like even though they were once installed just as a dependency of another package. You can mark such a package as manually installed by using apt install foo or apt-mark auto/manual foo. Packages which you have installed explicitly/directly via install are also never proposed for automatic removal.

**Editing Sources**

There is a cool new feature of *apt* that allows editing package sources, i.e. "sources.list", in your preferred editor while providing basic sanity checks:

# apt edit-sources

See [SourcesList](https://wiki.debian.org/SourcesList) on how to configure your sources.

Low-level Tools

Largely dpkg and friends, these are commands that perform a single task. They are most directly comparable to Redhat's "rpm" command.

* dpkg-reconfigure locales to reconfigure [Locales](https://wiki.debian.org/Locale).
* dpkg-reconfigure exim to reconfigure the exim mail-transfer agent. This command isn't currently part of dpkg; it's provided by debconf.
* dpkg, (see [DPkg](https://wiki.debian.org/dpkg)) invoked with various command-line arguments. If you find a place to download debian packages that is not accessible via apt, you can download the packages and install them directly. Note that this method will not automatically resolve dependencies between packages etc, therefore use apt or dselect if possible.
* dpkg --list-files textutils to list all the files provided by the textutils package. This only works for packages which are already installed. This can be useful to find out what form a package's documentation has been provided in.
* dpkg -S /path/to/some/file to find out which package a file belongs to.
* deborphan can be handy but is no longer required to recover space because apt now automatically uninstalls unneeded packages that were installed only to satisfy a dependency of a package that was specifically requested to be installed but has since been removed. It gives you a list of library packages that no longer have higher-level packages depending on them.

Contents of a debian file

To see the contents of debian packages in the Debian repositories, whether installed or uninstalled, install the apt-file package and enter

$ apt-file update

$ apt-file list foo

To get the contents of the file "abc.deb", you have to enter

$ dpkg-deb -c abc.deb

Reconfiguring of debconf

To reconfigure debconf so that it informs you of most changes of the new packages:

# dpkg-reconfigure debconf

Chose "medium" here

Package Management with apt-get and dpkg

**Package resource list for APT**

The file /etc/apt/sources.list is the definition file of the sources for apt. For further information about this file type

$ man 5 sources.list

**Retrieve new lists of packages:**

To fetch the new lists of all the packages from sources.list enter:

# aptitude update

or the equivalent

# apt-get update

This command checks for new release-files on the given servers. Luckily the command is clever enough to check if the release-file has changed after the last update.

**Upgrade of all the files:**

To make an update of all the changed packages, enter the line

# aptitude safe-upgrade

If you want to do this from cron you should consider using the cron-apt package.

The older apt-get based method is:

# apt-get upgrade -u

The additional flag -u shows a list of upgraded packages as well. If you want to run the update in a cron-job, you should use the flag -dy. With this flag, the new packages will be downloaded but not installed. You can install them later by writing apt-get update -u.

**Distribution Upgrade of all the files:**

**WARNING**

Following the upgrade instructions found in the [release notes](https://www.debian.org/releases/stable/releasenotes) is the best way to ensure that your system upgrades from one major Debian release to another (e.g. from jessie to stretch) without breakage!

These instructions will tell you to do a dist-upgrade (instead of upgrade) at least once. You would have to type something like

# apt-get dist-upgrade

at some point during the major release upgrade.

A dist-upgrade will also be required to keep up-to-date with the latest version in testing and unstable. A dist-upgrade may be very occasionally needed for you to obtain security upgrades in stable releases too in the case where packages change names.

**Discover a package:**

How can you discover which package to use without the GUI programs?

# apt-cache search sylpheed

Or use a web interface:

<https://packages.debian.org/search?sylpheed>

With the web search interface you can search the package contents for specific files:

<https://packages.debian.org/search?suite=default&section=all&arch=any&searchon=contents&keywords=bin%2Fsylpheed>

**Add a package:**

How can you add a package to your system without the GUI programs? All you have to do is find out the name of this package and enter the command

# aptitude install sylpheed-claws

or

# apt-get install sylpheed-claws

to add the package sylpheed-claws.

**Remove a package:**

To get rid of the package foo which is no longer needed, enter the command

# aptitude remove foo

or

# apt-get remove foo

Removing a package removes all packaged data, but leaves usually small (modified) user configuration files behind, in case the remove was an accident. Just issuing an installation request for the accidentally removed package will restore its function as before in that case. On the other hand, you can get rid of these leftovers by calling purge even on already removed packages. Note that this does not affect any data or configuration stored in your home directory.

So, technically if the package is needed by other packages you will be prompted to remove this package and all packages which depend on it. Also just removing a package does not remove the configuration files that relate to the package. To get rid of the package foo which is no longer needed along with all configuration files and related dependencies, enter the command:

# apt-get purge --autoremove foo

You can also use the command below to purge configuration files after issuing a normal by apt-get remove foo:

# apt-get purge $(dpkg -l | awk '/^rc/ { print $2 }')

**Warning**

Careful while removing packages. Pay attention to apt messages.

**Forcing removal of a package:**

While making a distribution upgrade it could happen that you have to remove a package first. But sometimes this is not possible with apt-get remove since the package has unmet dependencies or is not installed completely. You can remove the package foo by using the command

# dpkg --force-all --remove foo

**Read the contents of a installed package:**

Sometimes you want to know which files are in the package foo. To get this information enter the command

$ dpkg -L foo

**Get the description of a package:**

You don't know what the package foo is for? Just write

$ aptitude show foo

or

$ dpkg -p foo

**Search the package of a file:**

Have you ever wondered to which package a given file belongs? To get this information (e.g. for /usr/bin/apt-get), start the command

$ dpkg -S /usr/bin/apt-get

and it will tell you that this file is part of the package apt.