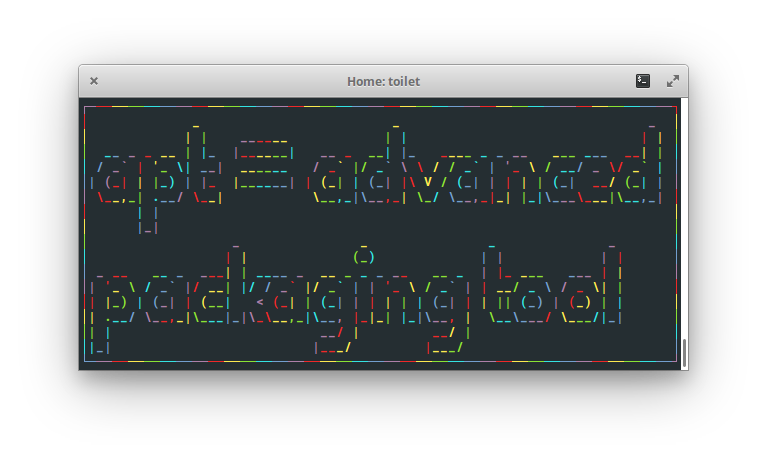
A Clear in-depth Guide to Apt

Prerequisites: Basic Terminal Usage Article Type: Informative

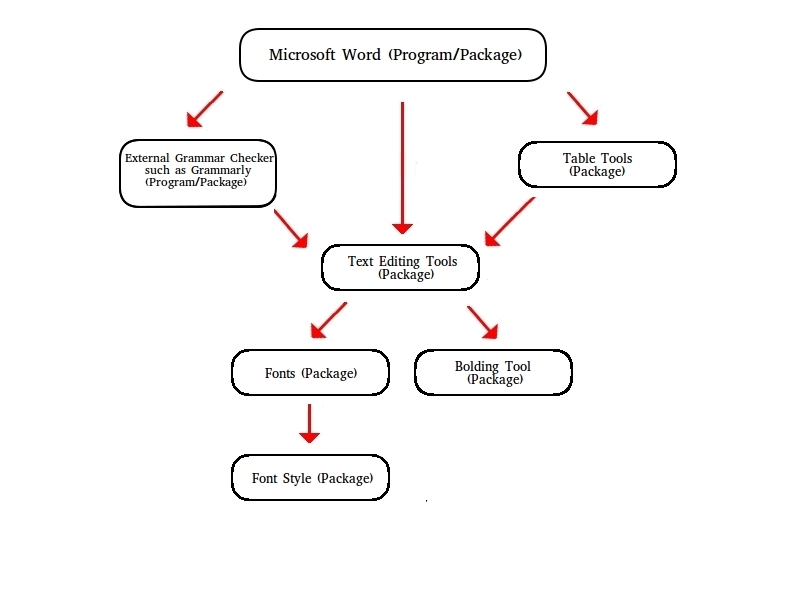


APT, or Advanced Packaging Tool, is a handy command line interface used for managing packages on Ubuntu-based Linux operating systems. It is the key to install your favourite software on your new operating system. Being one of the most used tools, there is certainly a merit to understanding how it works and the functions it offers. After learning the functions of **apt**, you will be able to install/uninstall packages, manage installed packages and explore other available packages.

Before we can understand how **apt** package manager works, we need to know what a package is and how it differs from a program.

A package is a bundle of code designed for one purpose. It could be used to modify a function in other software, or give your operating system a completely new functionality. As you can see from this definition, a program can be considered a package; it just has to be a bundle of code.

The definition of a package is simple, but the way of using it is slightly complex. Packages are related to one another and they cannot be considered as an independent piece of software. Let' use Microsoft Word as an example. It contains tools like bolding a word, changing a font, and changing font size. Each of these tools can be considered a package, or a sub-package of Microsoft Word as they are indeed a chunk of code for a specific function. We can even consider Windows as a package, which in this case, Microsoft Word would now be considered a sub-package. As you can see, what is considered a package can be very broad, but it is important to understand that packages depend on one another to function, and some of them can be treated as their own programs. The picture below summarizes the structure of a simple package quite well

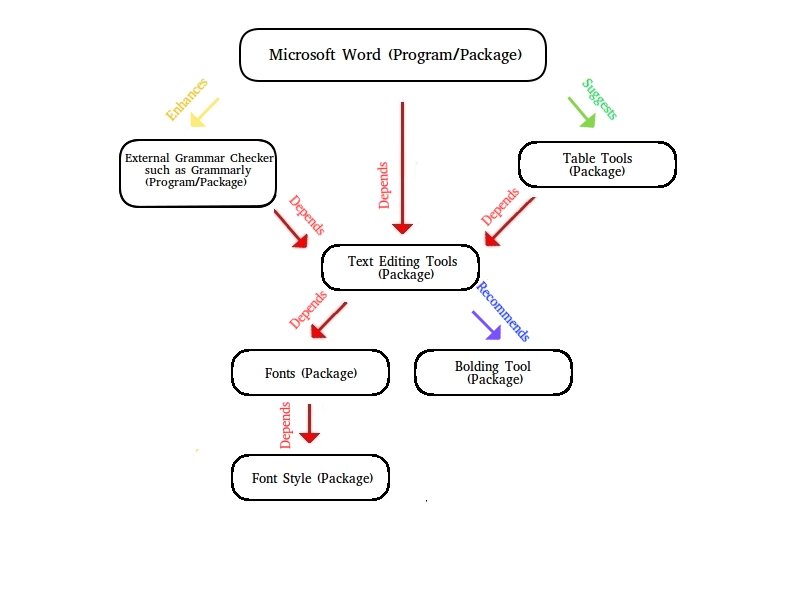


A possible implementation of the Microsoft Word package structure

There are four types of sub-packages: dependent, recommended, suggested, and enhanced packages, which are briefly explained in the table below. All examples of sub-packages are relative to the Microsoft Word example.

Types of Subpackages Explained

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Dependent | Recommended | Suggested | Enhanced |
| Definition | A package without which the program would not be able to function | A package that the program highly depends on but does not require it to function | A package that may be useful to the program but it is perfectly fine to run the program without them | A package that can add or enhance a functionality, usually in a way that the program is not originally designed for |
| Examples | Fonts, Save tool | Bold, Underline Tools | Table Tools  Bibliography Tools | External Spell Checker Tool such as Grammarly |



A possible implementation of Microsoft Word’s package structure along with their relationships

When you download a package, you do not have to download all the dependencies separately. Fortunately, **apt** is very smart and it will automatically install all dependencies for the program you desire.

All your packages installed come from a software repository, or repo for short. You can consider a repo as a cloud, similar to Google Drive. Each repo contains packages that were created for a specific purpose. Several repos come with a fresh-installation of Ubuntu, however, to install more diverse software with **apt**, you may have to add new repos.

**apt** manages its available packages for download in a local database. Each entry in the database references an available package in a linked repo. This helps **apt** perform its functions faster as the list of packages are in one place. However, most of the time the database becomes outdated, as updated or newer software often gets added to the repos. Thus to work with the latest software, you must update this database regularly. This is called updating your package manager. You must also update your package manager after you add a new repo to reference the packages in the new repo.

Basic **apt** commands:

To update your package manager, use:

sudo apt update

This will update **apt**’s database on the packages available for installation.

To install a package, use:

sudo apt install <package>

You will need to know the package name to install it. For example, use *sudo apt install firefox* to install firefox. If you install a package that you have already installed, it will update the package for you.

To update all your packages, use:

sudo apt upgrade

sudo apt full-upgrade

”apt upgrade” will ensure all your packages are up to date. ”apt full-upgrade” works the same way as apt upgrade, but it will also delete packages that are required for an update, whereas “apt update” won’t do that. For both commands, **apt** will only upgrade the packages you installed with apt. If you have installed programs through .tar files, you will have to upgrade them yourself.

To remove a package from your system, use:

sudo apt remove <package>

sudo apt purge <package>

”apt remove” will remove the package without deleting the configuration files. With this way, if you reinstall the package, your settings will remain intact. ”apt purge” will remove the package as well as the configuration files.

To remove unnecessary packages, use:

sudo apt autoremove

Some packages are auto installed to satisfy the dependencies of another package. You will need to use this command to delete them, as these packages will not be automatically uninstalled even if the package requires uninstallation.

To search for packages, use:

apt search <search\_term>

This will list all packages that match with your search term.

For a list of installed packages, use:

apt list --installed

To see the details of a package, use:

apt show <package>

To learn more about apt, use:

apt –help or man apt

You may have seen other package managers such as apt-get, pip and dpkg. There are many package managers used in the world, where each of them has its advantages. For example, pip is a package manager for python and dpkg is a lower level package manager. Although this article does not directly teach you how these work, package managers generally work the same. Being able to apt means that you can easily learn how to use other package managers. Best of luck in your Linux journey!

Resources:

If you want to learn more regarding the relationships between packages, check the link below:

<https://www.debian.org/doc/debian-policy/ch-relationships.html>, Debian

For more information on the apt commands, check:

<https://www.howtogeek.com/63997/how-to-install-programs-in-ubuntu-in-the-command-line/>, How to Geek