Simple package management with Synaptic

A package management GUI for Debian-based distributions

Marco Marongiu

If you don't like using the command line, and you want to manage your program installations without typing a command, then read on: this article is for you!

Ladies and Gentlemen: Synaptic!

Synaptic is a graphical user interface (GUI) for managing software packages on Debian-based distributions. If you are using Debian or Ubuntu you will easily find Synaptic in the System Tools menu or in the Administration menu. Synaptic uses the GTK graphic libraries (GNOME's ones) . So, if you are using GNOME on your debian-based distro you will probably have Synaptic installed as well.

To manage package installations you need administration privileges; so you need to either be root or to authenticate as such. Normally, you will see a window like the one in figure 2 where you can type in the root's password.

Exploring Synaptic

Once Synaptic has started, you will see an interface like the one shown in figure 3.

You have a menu on top, then a panel with a few buttons, whose functionality will become clear as you read further into the article.

On the left, there are four buttons at the bottom that will determine what is shown in the menu above them. In figure 3, the "Status" button is pressed; so, you can select the packages you see in the list by status. If you select "All" (like in the figure) you'll have a complete list of available and installed packages. "Installed" shows just the installed ones and so on. The right side of the window is divided into an upper and a lower portion; there is a packages list in the upper part and when you select a package in the list, you get a description of it in the lower part.

Packages can also be grouped by functionality (e.g. text editors, documentation, tools for managing images, etc.). Just press the "sections" button and you can select the packages you like among a number of different sections.

Figure 1: Depending on your GNOME version, you will find Synaptic in the Administration menu or in the System Tools menu

Figure 2: You need adiministration privileges to run Synaptic

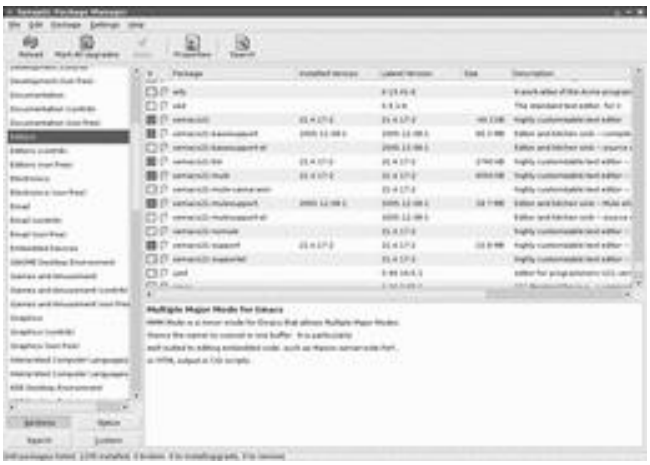
Let's work!

Now that you have a basic idea of the tool you have in your hands, it's time to go deeper and see how to use it. The first thing you probably want to do is install something. However, it's better to use a little patience and do first things first.

Synaptic is a graphical user interface for managing software packages on Debian-based distributions

Figure 3: Once Synaptic starts, this is what you'll see

Figure 4: Packages can be selected by functionality using the "Sections" button



As you may know, programs are made by human beings, and human beings aren't perfect; therefore, a number of bugs are found every day in computer programs, and free software is no exception. The difference with free software is that very often patches are out a few hours or days after a security problem is spotted, and you'd better apply them.

Keeping the system safe

So, the thing you should always do as soon as you launch Synaptic for the first time in a day is update the package information from your repositories and see if there are any packages that need a security upgrade. On the command line that would mean issuing a couple of commands; in Synaptic it's just a matter of a few clicks. First, you have to click the "Reload" button, which updates the information of the available packages. Then, pressing "Mark All Upgrades" will automatically select all the packages that need an upgrade. Finally, press "Apply" and confirm you really want to upgrade the selected packages. After a while you'll see all the security patches applied and you can safely do all other package management operations.

Installing new software

Now I'll show you how you can install software with Synaptic through a practical example. Since I have an MP3-enabled car stereo, I rip my CDs so that I can listen to them while driving. That way, I have only a few disks to carry

Figure 5: Downloading the upgrades

instead of having the car filled with them. In doing this, I became interested in MP3 streaming a-la-shoutcast. So I decided to install the Icecast (`http://www.icecast.org/`) software (a free software alternative to shoutcast) and test it. If you want to have more information about Icecast, you can look it up using the "Search" button; after you find it, you can select it and press the "Properties" button to examine it in depth.

Installing a software package with Synaptic is just a matter of a few mouse clicks

Installing Icecast with Synaptic is simple: just look it up in the packages list (by hand or using the "Search" functionality; right-click on it and then select the "Mark for installation" menu item (see figure 7).

You will be shown a list of packages, in case Icecast needs other packages that are not installed on the system yet; this situation is technically called a "dependency" (see the text box entitled "Dependencies and conflicts" for more information). You are asked to mark the other packages for installation as well, or to abort the installation (step 1 in figure 8). In my case, I just needed one more package: libcurl3-gnutls. I accepted it, applied the changes, and was asked to confirm again the list of packages to be installed (step 2).

Default upgrade or smart upgrade?

Depending on what you are doing, when you ask for an upgrade you may be shown a window like the one in figure 6.

Figure 6: default upgrade or smart upgrade?

If you are using a plain, vanilla distribution (you haven't installed any packages that come from repositories other than those of your distribution) you shouldn't need anything more than a default upgrade. You definitely need a smart upgrade when you are changing your distribution version (for example, if you're upgrading from the old Debian "Woody" to the newer "Sarge" release). Otherwise, doing a default upgrade should always be enough.

Figure 7: Selecting Icecast for installation

Figure 8: Installing Icecast in three steps

At this point you just have to look at the text window (step 3) to see what's going on and close it when you are done!

Of course, you still don't have your own home-made net radio, but that would definitely be the subject of another article. So, I'll leave you alone with that (or you can try and install another package of your choice that is more useful to you).

Removing installed software

If you are a Linux novice, you probably want to experiment a lot with it. This often leads to installing tons and tons of different packages. You play with them and then forget all but a few of them. You have a big hard disk, so this may be not a problem at first. But, *big* doesn't mean infinite, and the time will come when you have to remove something.

Nothing will happen until you hit the "Apply" button

Synaptic comes in handy here, too. And since it happens that I have to remove the Skype package from my distribution, I'll show you how to use Synaptic to remove a package. In the beginning, I was happy that the Skype people offered a package repository for their software, so that I could keep it up to date. The deb package for Skype is expressly made for the Debian "Sarge" distribution, and in the beginning it installed okay on the testing distro too. Unfortunately, that came to an end and some conflicts eventually arose. I then had to remove the Skype package and install it using other means.

As with before, look up Skype in the package list; then, right-click on it and select "Mark for Removal" if you just want to remove the software, or "Mark for Complete Removal" if you want to remove both the software and all configuration files (see the textbox for details about configuration files).

Dependencies and conflicts

Some packages have dependencies. If you need to install one of these packages—for example, called *package A*—on your system, you are first required to have installed any other packages which *package A* depends upon. These other packages are referred to as "depenencies" of *package A*. For example, the package of word processor will need packages that contain fonts to work properly.

Conversely, if you have *package A* and *dependency 1* installed already, and you want to remove *dependency 1*, what happens? Since *package A* needs *dependency 1*, chances are that *package A* will not even work any more. This situation is called a "broken dependency", and synaptic helps you avoid such nasty situations by warning you of what you are doing and asking you to remove *package A* with *dependency 1*.

In opposition to the concept of dependency you have the concept of "conflict"—a relationship which means one package requires other packages to **not** be installed in order to work properly. In other words, if *package A* will not work when *package B* is installed, *package A* and *package B* are said to be in "conflict". For example, in my distribution the package `gaim` (an instant messenger software that supports AIM, ICQ, MSN and about half a dozen of other protocols) conflicts with two other packages (`gaim-gnome` and `gaim-common`). This means that if you try to install `gaim` and either of the two conflicting packages (`gaim-gnome` or `gaim-common`) are installed, Synaptic will ask you to remove them.

Configuration files

Programs can behave differently depending on certain variables; for example, when your graphical session starts it needs to know certain things, e.g.:

- screen resolution
- background image
- whether or not to allow the administrator to log in

This information is stored in and taken from "configuration files" (files that are read by programs upon startup and that tell them how to do certain things). System-wide configuration files are almost always in the `/etc` directory or in a subdirectory of `/etc`. If you need an example, and you have a configured network card in your PC, have a look at the file `/etc/network/interfaces`.

Then, as before, click "Apply" and confirm that you really want to modify the system the way you asked, and again watch the text window to see your installation of Skype go away. That's all you need to do!

Wrecking the system (ack! NO!)

In this particular case I was lucky and removing Skype was all that was needed. But, what if you are removing a package that is needed by other packages?

Suppose that you've decided to clean up your system and remove a ton of packages that you installed a long time ago and don't need anymore. Unfortunately, you don't remember exactly what you did install when you experimented, and you resolve to review all of the installed packages with Synaptic. You read the description of each package, one by one, and you hit a package named "mypackage" whose description is quite obscure to you: you decide to wipe it out, and mark it for removal.

Synaptic stands for a while, consuming a lot of CPU, and then it displays a window like the one in the figure 10. So what's going on?

Since removing "mypackage" would break any packages that are dependent on it, Synaptic is asking you for confirmation to remove those packages as well. Don't panic! Nothing has happened yet. You just have to review the list of packages to be removed and decide if you want to proceed or not. In case you accidentally hit "Mark", don't worry: nothing will happen until you also hit the "Apply" button. So, even if you went wrong the first time, you still have a chance to undo the unwanted changes. If in doubt, just close the program abandoning all changes and start all over again.

Figure 9: Selecting Skype for removal

Figure 10 was obtained pretending I was removing the `libc6` package, which is **fundamental** to the system. Synaptic stood about one minute eating CPU cycles before providing me with the list of packages to remove (nearly the entire system!). If you want to experiment you can do the same thing, possibly with other packages and without removing anything, just to be safe.

Figure 10: Are you sure you want to remove that package, and all packages that depend on it?

What are packages, anyway?

I've shown you nearly everything you need to know about Synaptic, but this article won't be complete without a quick peep behind the scenes. If you want to get the most from Linux you should get to know more about what goes on under the hood of your graphical interface. In this case, that means having at least a basic idea of what a packages really is.

In modern Linux distributions, bundled software is organised in packages. You can think of a package as an archive file composed of:

- a name for the package
- a number of files that are part of a piece of software
- a number of scripts that help the system manage installation, configuration and removal of the software
- information about the dependencies of the package (e.g. dependency information in the xchat-common package states that, if you install it, you will be asked to install xchat as well, otherwise the files in xchat-common would be pretty useless...)

Two formats are quite popular at the moment: the RPM and the DEB. RPM stands for Redhat Package Manager, and was created by the people at Red Hat. DEB was created by the people of the Debian Project. I am not going to delve into the details of the package format or start a religious war about which format is better than the other. Suffice it to say that, together with the attention to detail of the Debian people, the DEB format is at the foundation of the robustness of the Debian distribution and is a solid base for all Debian-based distributions, like Ubuntu.

Package management in the old days

In the early days, package management in the Debian distribution was the business for two tools: `dpkg` (low-level, command line interface) and `dselect` (console-based interface). While I use `dpkg` from time to time, I never loved `dselect`. But I had nothing else...

Then came APT (the Advanced Package Tool) and `apt-get` (a higher level command line utility). While I had to be careful about package dependencies when using `dpkg`, `apt-get` did all the dirty work automatically. In other words, if you're installing a package, it will ask you to install the dependencies as well; if you remove something, it will warn you that you are going to make other packages unusable and proposes to remove them as well. Applying the latest security patches is just a matter of issuing a couple of commands; and the same holds for upgrading the whole distribution to a new release

So, the bad times were over: I could finally give up `dselect` and rely on `apt-get`. But if you're not a command-line junkie you will probably find this solution suboptimal as well. That's why we have Synaptic!

Summing up

Having read this article you should now know how to use Synaptic to look for packages and install or remove them. You've also had a quick peep behind the scenes so you should also have a basic understanding of what's actually happening. Now it's time for you to experiment. Have fun and enjoy Synaptic!

Copyright information

About the author

Born in 1971, Marongiu graduated in applied mathematics in 1997; he's now a full-time system administrator for a European ISP. He's also a Perl programmer and technical writer and lecturer by passion, and is interested in web and XML related technologies. Marongiu has been a Debian User since version 1.1.10 and he helped found the GULCh Linux Users Group (Gruppo Utenti Linux Cagliari), the first one in Sardinia. He recently became a father to his first son Andrea, and he's been trying to reorganise his life since, so that he can start writing technical articles again and holding seminars.