**The Linux Debian package components structure**

Debian is known to be upgradable at any time without a reboot of the system, even from a remote terminal. A Debian system is usually only installed once in its lifetime, except the root disk suffers from hardware damage and requires a new installation. In other words, Debian systems don't get installed every now and then but upgraded to the current stable or unstable distribution instead.

**Package Basics**

Before we continue we need to draw attention to the basics of packages and management. In order to help maintaining the GNU/Linux system you have installed the Debian project splits the installed software into small partitions called packages. These packages contain the actual data or software that you install on your system.

With packages it is quite easy to install certain parts of the system and leave away others. For example, if you want to used KDE but not GNOME you don't have to install GNOME. However, thanks to sophisticated dependencies should a KDE program require a part of the GNOME system, this part will be installed as well, but only the required part.

The project distinguishes between source and binary packages. The binary packages are what the user or administrator will see most of the time. They contain pre-compiled software which is installed on a Debian system.

One binary package is always associated to exactly one source package, which — as the name suggests — contains the source for said binary package. In order to provide more sophisticated dependencies several source packages build more than only one binary package.

**Static Package Information**

A binary package consists of more than just a set of files that are installed on the system. Next to the installed data are meta-information about the package and its relationship to other packages in the distribution.

This information is displayed with apt-cache show package anddpkg -I package.deb respectively if the package is locally available already. These parts are used by the packaging system to determine from where to fetch a package, which other packages are required, which other packages must not be installed at the same time, which other packages may be overwritten and so on.

Most of the information is provided by the maintainer of the given package and is subject to permanent adjustments during the lifetime of the package. Especially when talking about libraries such meta-inforation will change with future updates.

**Conflicts**

This information is not used very often since most packages can be installed and run along with others. However, some packages are conflicting with others since they provide the same files or allocate the same port or provide the same service. These can't be installed at the same time. This field will tell the package manager to abort with an error message or to remove the other package so that the new one can be installed.

When using the unstable distribution, it is possible that you notice apt-get wanting to remove half of the installed system. This often originates in a library transition that includes a new conflict. Before the other packages are recompiled to use the new Library, they are part of the conflict. Usually it is sufficient to just wait a few days until they have been uploaded again and are compiled against the new library.

For example, there can only be one mail-transport-agent like Exim, send mail, postfix etc. or there can only be one development package of a library (contrary to the runtime package that may be installed with several versions).

Replaces

This field defines that the new package replaces (parts of) another package. With this definition it is possible that two packages that contain the same files may be installed at the same time.

This is often used when files are moved from one package to another, so that the package manager can install the new versions even though there are some file conflicts.

Provides

This field is used for virtual packages and when a package is renamed. It basically tells the package manager that this package provides another package. In the case of a virtual package the semantic is that this package provides a certain functionality that other packages may depend on.

In case of a real package, i.e. when another package was renamed, it means that the package manager may remove the other one if it also replaces it at the same time.

Package Managers

The packages on a Debian system are maintained by dpkg. This is the package maintenance system. There are several frontends to this system. The most commonly used and most sophisticated one is apt-get. However, the older frontend dselect still exists and is in use by those who are used to it.

A more up-to-date text-based frontend to the package maintenance system is aptitude which is used to upgrade from one distribution to another these days. Those who prefer graphical programs can run Synaptic which enables you to install, upgrade and remove software packages in a user-friendly way. Finally, there is also kpackage that can cope with Debian packages.

Installed Packages

There are two lists of installed packages available on Debian. The original file these lists are created from is /var/lib/dpkg/status. This file must not be corrupted, or otherwise your system is hosed. This is the main database for the package manager dpkg.

The Debian package system keeps an older copy from the last but one dpkg run in /var/lib/dpkg/status-old. In order to preserve the system for greater damage upon a crash or filesystem corrupting a daily backup of this file is created into /var/backups when the file differs from the last copy. The backup code is in /etc/cron.daily/standard.

The informal list is created by the command dpkg -l or by starting deselect and using the Select item. Or by using any of the other package manager frontends. This list consists of the name of the package (stripped down to some 40 characters), the installed version of each package and a short description. This is intended to be human readable.

For technical purpose you can make dpkg to generate a list of packages and their selection status (i.e. install, hold and deinstall). This is created by dpkg --get-selections. This output is intended to be parsed by a program again, such as dpkg --set-selections which will change the status of packages in its internal database. The dselect-upgrade action from apt-get will install all new packages afterwards.

**Information about Packages**

As mentioned above each package consists of installed files and metadata. To display a part of the package information you have to issue dpkg -s package. Among others you will see a description of the package as well as dependency information and information about configuration files used by this package.

The mentioned configuration files will not automatically be overwritten on an upgrade if you have modified them manually. You will be asked if you want to overwrite it or keep the old file. However, on an upgrade if the file was not modified and the package comes with a new version it will be overwritten.

**Locate Files and Packages**

One advantage of using package management to install packages is that it keeps information about files and packages somewhere on the local system. The benefit is that you can map any file on the local system to a package and display status information.

To find out which package contains a given file, issue the command dpkg --search file or dpkg -S file. If you are able to specify the absolute path, do that, if not dpkg will display all packages and files it has found.

Since this method only works for installed files and packages you'll have to issue a different command when the file is not installed on your system or the package isn't. This is useful especially if you are looking for a package that contains a command you only have about.

The Debian project provides a web-based search tool to look into their packages databases. On [debian.org](http://www.debian.org/distrib/packages) you can search for package names and descriptions as well as for files within any package that is distributed by the Debian project.

Debian provides a file that contains a list of all files of the distribution. It is called Contents-$arch.gz and is found on their FTP-server in /debian/dists/$dist/. Since there are differences among different architectures (i386 for example has lilo, sparc has silo etc.) there is one file for each architecture.

If you want to use a current version of that file, there is no need to fetch it every day. Due to the time requirements for such a file to get regenerated (approximately one hour per file) these files are only generated once per week. Thus, it is sufficient to update your copy once a week. New versions of packages often don't imply a change in the Contents files, but only entirely new packages or splitted/merged packages. If you don't need such an up-to-date copy, for the unstable distribution, it is usually sufficient to make a sporadic update or update that file once per week.

The Contents files are plain text files, so you can simly use grep or zgrep respective to locate a file or package in it. There is also a very simplistic frontend that makes use of this and is also able to update the file. Since the author uses it for the unstable distribution that distribution is hardcoded in it. Check out [findpkg](http://www.infodrom.org/Infodrom/tools/findpkg.html).

**New Packages**

There exist a couple of ways to install new packages on a Debian system. First of all, the package should be in .deb-format. If it is not and it is a binary package, try using alien to convert it.

If you have already fetched a .deb-file you can install it with dpkg --install file.deb or dpkg -i file.deb. The package manager will then check all dependencies and install the package if no problems occurred. You can add more packages to the command line. If the dependencies are not fulfilled it will bail out.

If the package does not yet exist locally, you may want to use apt-get install package. APT will check if all dependencies are fulfilled and will download the other required package as well. When there are conflicts, it will try to resolve them, which may result in the removal of other packages.

APT stores these files in /var/cache/apt, so if the connection goes down in the middle of the download, just issue the command again. If you add -d to the command line, APT will also only download the packages but not install them. If you add -s, APT will simulate the installation and display some information.

Even if it may be a little bit confusing, to upgrade a package to the newest version in the Debian archive, you also have to use apt-get install package. APT will automatically use the most recent version available according to its internal database.

**Package Database Update**

Both dpkg and APT store information about available packages on the local system. Before you can access newer packages from Debian archives, you'll have to update the internal database against updated sources. To update the internal database of available packages, issue the command apt-get update.

All package sources are configured in the sources.list file. After a new installation this file is initialised with proper sources, including the installation CD or DVD. When you don't need this anymore, just comment out the respective line before executing apt-get update.

If you use a Debian archive provided on the Internet this will usually download several megabytes of index files. You will have to issue this command more frequently if you use the testing or unstable distribution instead of the stable one.

Subsequent calls of apt-cache show will show both the installed and the new version of a package if it has been updated in the meantime. All new installations of packages refer to the new database and hence will always install the most recent version of a package.

Even though you will normally not need to know the details, there may be situations in which they are helpful. APT saves the entire package index files it has downloaded from the Internet. They are placed in the directory /var/lib/apt/lists. This directory also contains descriptions and their translation into other languages.

APT Sources

Before giving an introduction into apt-get (8) I want to provide some information about its sources first, since several frontend programs (like dselect and capt) are also using this information. You will need this information if you want to direct these programs into the proper archive direction.

Sources for APT refer to the file /etc/apt/sources.list which is a plain text file containing resources for apt. These resources point to a directory that contains .deb files and provides a Packages file. Usually you will use official Debian archives, but you may also use unofficial archives. Each line can point to a binary cd, an HTTP- or FTP-resource or a local mirror.

For binary packages you should only use one of stable, testing and unstable. These three distributions represent sets of packages that Debian maintains as a set. As a regular user you should only use the stable distribution, which is the best tested and released distribution that Debian can provide. If you want to run bleeding-edge software or require to run most recent software, you may have to use unstable distribution. In that case please be warned that unstable may break things from time to time.

There are a couple of other non-official resources maintained by Debian developers or people who want to distribute non-official packages, e.g. nightly builds from CVS. Stephane Borzmeyer maintains a [list](http://www.internatif.org/bortzmeyer/debian/apt-sources/index-list.html) of unofficial sources.

**Accessing network servers**

This is the most often used method for apt-get, accessing a server located somewhere on the internet and using the Debian archive there. You'll have to specify the URL until the dists directory occurs (here: http://http.us.debian.org/debian). The next argument is the distribution to be used, this is the path component until the section (main, contrib etc.) occurs (here: stable). Finally, you have to specify all sections (here: main). See the example below.

deb http://http.us.debian.org/debian stable main

**Accessing local directories**

For local directories, not much has to be changed. The URL access method has to be changed to file:. The next component is the path on the local system until the dists directory occurs (here: /mirror/debian).

deb file:/mirror/debian stable main

**Accessing cd-roms**

CD-ROMs require special handing since they are not available all the time like ftp servers. Even worse it is quite unlikely that there is always the same cd-rom inserted in the drive. To cope with this situation apt-cdrom(8) was invented. When called with the add command it will umount the current cdrom and then ask the user to insert the Debian cd.

apt-cdrom add

The program will add proper lines to /etc/apt/sources.list so that later calls to apt-get will know about the availability of these cd-roms. apt-cdrom will take care of determining the structure of the cd as well as correcting for several possible mis-burns and verifying the index files. apt-get will not only add the cdrom to the list of APT resources but also scan them and add their index files to the internal database of apt. Thus, you must not create the resource lines by hand.

If you don't want to use the cdroms anymore (e.g. after installation is done and further updates should be used from the internet), you have to remove the respective lines from /etc/apt/sources.list. These lines look like the following:

deb cdrom:Debian GNU/Linux 2.2 r0 \_Potato\_ - Official i386 Binary-2 (20000814)]

If you want to use a different set of cdroms, you have to remove these lines from /etc/apt/sources.list first and then call apt-cdrom add again. This will scan every new cdrom and add proper entries to the internal databases.

**Configuration for stable**

For the stable distribution (codenamed potato as of this writing) there should be lines in /etc/apt/sources.list like the following, given that you want to use the archives from the internet. If you also want to use non-free software, simply add "contrib non-free" to "main".

deb http://http.us.debian.org/debian stable main

**Configuration for unstable**

For the unstable or testing distribution the lines look similar:

deb http://http.us.debian.org/debian unstable main

**Package Difference Files**

Beginning with Debian 4.0 (codename etch) APT support package diff files and defaults to downloading them instead of full index files. They contain only the differences between two versions of the package index file Packages. When updating frequently this will save a lot of bandwidth since the differences are a lot smaller than the entire files.

The default setting is to download only differences of package index files instead of the entire file. This behavior can be altered which may be useful when the administrator doesn't update too frequently or when there is a Debian mirror in the local network. To switch back to downloading the entire index files add the following to /etc/apt/apt.conf.

Acquire::Pdiffs "false";

If you would like to adjust this setting only temporarily, you can do so on the command line as well without having to edit a configuration file. apt-get supports configuration on the command line with the -o argument. The following command will update the internal database against the one provided by the Debian project on the Internet and temporarily disable fetching diff files:

apt-get update -o Acquire::Pdiffs=false

**Introduction to dpkg**

The main package manager on a Debian system is dpkg(1). It handles installation, upgrade, configuration and removal of packages as well as dependency handling. These days it is not called directly normally but through a frontend like apt-get, deselect or capt.

Never the less it is always good to understand that the actual work behind the scenes is done by dpkg and that you may always call it manually if you should run into trouble the frontend cannot cope with. After the most important commands you'll find a list of important options to add. For a comprehensive list of arguments, please call dpkg --help

**dpkg --install**

Add an arbitrary number of package archives (i.e. .deb-files) to the command line. dpkg will check their dependency information and install the packages if there are no problems. When there are problems, you will have to fix them manually. This command may be abbreviated with -i.

**dpkg --configure**

With this command you are able to configure packages that failed to configure in the first stage. This usually only happens when you upgrade to the current unstable distribution and install broken packages that don't configure (i.e. whose preinst/postinst scripts produced a failure). dpkg will retry to configure the specified packages. If -a is supplied instead of a list of packages dpkg will check its database for any unconfigured packages and try to configure them.

**dpkg --list**

Add an arbitrary number of package names to the command line. The program will display a short status for each package together with their name, version and short description. If no packages were provided dpkg will display information about all installed packages. This command may be abbreviated with -l. The output looks like:

**dpkg --status**

Add an arbitrary number of package names to the command line. dpkg will display the status and description for each supplied package. The output also contains information about registries configuration files for that package. You may want to use this information if you want to change the configuration of a package. This command may be abbreviated with -s. If you don't need information referring to the local installation of the package but only want to read meta data you may use apt-cache show instead since it is faster.

**dpkg --search**

Expand the command line with a file or directory substring. dpkg will search in its database for that string and display all files that contain this substring together with the package which contains the file. This command may be abbreviated with -S.

**dpkg --listfiles**

Using this command you'll see which files are contained in a particular package. The output is the same as generated by tar (1). This command may be abbreviated with -L.

**dpkg --remove**

Add an arbitrary number of packages that you want to remove from your system. dpkg will only remove these packages that don't leave packages with broken dependencies. Thus, if you want to remove a library another package depends on, you'll have to remove the other package at the same time or before. This command may be abbreviated with -r. If you have to remove a package regardless of other packages with broken dependencies, please continue and check the option --force-depends.

Please keep in mind that removing a package does only mean that binaries and variable files are removed but no configuration files from that package. If you want to remove the package entirely, i.e. including configuration files please read on and use --purge.

**dpkg --purge**

This command will remove all packages specified on the command line together with their configuration files and meta data. dpkg will only purge the packages that won't result in other packages with broken dependencies. This command may be abbreviated with -P.

**Option --force-overwrite**

For the released stable distribution this flag is turned on by default, though it's turned off for the unstable distribution. If not turned on dpkg will break if a package is to be installed that will overwrite a file from another package while the new package does not officially replace the other package (by using the Replaces: information).

For the unstable distribution this is likely to happen for all packages that have to interact with other packages that contain the same files.

**Option --force-depends**

This option must not be used if you don't know what will happen. It is only documented here because it may be useful to cope with unstable and broken packages. You should not require its use on a system that runs the stable distribution.

If this option is used, dpkg will turn off its dependency check. This will most probably result in broken packages or packages whose dependencies are not satisfied, though some components may still work. This may be the only way to remove a broken package in order to install a fixed package.

When you force to install or remove a package with this option, apt-get will most probably not work anymore until you have fixed the problem.

**Option --force-conflicts**

You should never have to use this option. It will bypass the conflict check of dpkg and will enable you to install two or more conflicting packages. This is most probably not what you want to achieve.

**Introduction to apt-get**

The program apt-get(8) provides the most convenient interface for upgrading and installing packages. For people who don't find dselect(8) too much userfriendly and don't mind typing a short command in a shell, apt-get is most probably the best choice for their package maintenance.

The program maintains its own set of index files that are stored in a binary way and accessed using optimized methods. Thus, accessing them works faster than similar messages from dpkg. However, it also means that if you are switching from apt-get to deselect you may have to re-transfer the index files again. If you are running the unstable distribution the index files will change about daily (except when our main archive server crashes and no new packages were processed) so, you'll have to retransfer them befor any other command as well.

Even if apt-get will perform important tasks, it is only a frontend to dpkg. Actual installation and upgrading of packages will still be performed by dpkg. However, apt-get is able to fetch files from the net, perform dependency checks and call dpkg with the correct order of packages.

If you notice that apt-get isn't able to finish downloads you can increase the number of retries that it should perform before it finally gives up. This is done by adding APT::Acquire::Retries=20 to /etc/apt/apt.conf.

If you want to see the urls for packages that apt-get would download, you'll have to add -y --print-uris to the command line (-y will bypass the first question). You can easily parse the output and create an input file for wget(1) or similar.

Please read all subsequent sections covering apt-get carefully before working with the program.

**apt-get update**

This command will upgrade the index files for all targets listed in /etc/apt/sources.list. Before actually downloading any file from the network it will check if it is newer than the copy already present if any. This will save you some bandwidth for less frequently updated archives.

On a host with good network connection on which the unstable distribution is running you may want to put that call into the crontab (5) for the user root.

**apt-get install**

This command will install all packages provided on the command line, given that they don't produce a conflict in some way. When these packages depend on others that are not yet installed, apt-get will download and install them as well. If that causes conflicts old packages will probably be deconfigured so everything will work well.

If files actually need to be downloaded then apt-get will display the amount of data that have to be downloaded and asks for the users permission. If the connection to the internet breaks down in the middle of the download session, just restart it, apt-get will cope with it properly. If you only want to download files but not install them, use the -d switch. All packages will be downloaded into the directory /var/cache/apt/archives.

If you specify packages on the command line that are already installed, apt-get will upgrade this package to the most recent version found in all the. So if you only want to upgrade a package, use the install command.

If you want to reinstall a package that is already installed in the most recent version, please add --reinstall to the command line. apt-get will then fetch the most recent version regardless of the installed one.

**apt-get upgrade**

With this command you will upgrade the entire system to the next revision. It will upgrade packages one by one if a newer version is available. This should only be used if you stay with the same distribution (i.e. stable, testing, slink, potato etc.). This method is used for security updates as well.

The Debian Project provides updates to a released distribution. These are called revisions and will overwrite the former release. Say potato was released in 2000 as 2.2r0 (implicit revision 0) the next revision would be 2.2r1. Updates to once released distributions will only contain security updates and eventually very few other updates like fixed packages if they were extremely broken. Security updates from [security.debian.org](http://security.debian.org/dists/) will be pushed into these updates.

Thus if you upgrade to the next release of a stable Debian distribution it is quite unlikely that you would download large amounts of packages or data. Though, if there were a lot of security updates, it can still result in a couple of megabytes.

Since the unstable distribution is target to a lot of changes and package reorganizations it is quite likely that a normal upgrade will not result in what you expect. Please consider running the dist-upgrade command instead.  apt-get dist-upgrade

This command will upgrade the entire system. In addition to what apt-get will do for a regular upgrade it will handle changed dependencies etc. in an intelligent way so that the upgrade is as smooth as it can be. This should be used if you don't update the unstable distribution on a daily basis, if you are upgrading from stable to unstable or if you are upgrading from one stable version to the next one (for revisions upgrade is sufficient).

apt-get has a "smart" conflict resolution systems that helps with upgrading the distribution. It will upgrade the most important packages first at the expense of less important ones if necessary. If you want to upgrade the packages as well that were left out you can run apt-get upgrade afterwards. This should upgrade all other packages for which a newer version is available and that don't produce a conflict.

**Configuring APT**

The main configuration of APT refers to the sources.list file in /etc/apt. This file contains a list of mostly network-reachable package sources to use. This is the basis for all package installations and the information displayed by the apt-cache command.

The behavior of APT can be tweaked in many other ways as well. The behavior is configured with the apt.conf in the /etc/apt directory or with files in the apt.conf.d subdirectory in the same directory respectively. When the configuration is split among different files the filenames have to start with two digits.

The format of these configuration files is modelled after a format the [Internet Software Consortium](http://www.isc.org/) uses for their tools such as BIND and DHCP daemon. All settings are organised in groups and partially even sub-groups. Lines starting with two slashes are considered comments and, hence, ignored. Each block needs to be terminated with a semicolon.

The general format for a configuration option is a key-value pair:

Option "value";

Acquire::Pdiffs "false";

However, since options are grouped and since the APT configuration format allows setting a scope for included options, the configuration files can be written in a nicer format. In the example above the Option Pdiffs is part of the Acquire group. In the single-line format options, groups and sub-groups are delimited by double colons. In the multi-line format groups and sub-groups embrace options by using curley brackets as shown in this example:

Acquire {

Pdiffs "false";

Retries "0";

};

This can be help keep the configuration file readable and easier to read since options that belong to each other are naturally grouped together.

The general rules for configuring APT is explained in the manpage apt.conf(5). However, only very few options are listed there. In /usr/share/doc/apt/examples you'll find all possible configuration options with valid settings in the file configure-index.gz.

**Translations**

Package descriptions and their translations are stored in separate files in order to keep Packages files small. Translations also have their source entirely outside of the package they describe. Since they are stored in a different file apt-get needs to be configured which language files to load during package database updates.

You can configure multiple languages, or just the one you would like to support. Later on programs decide on the environment variable LC\_MESSAGES which language to display. The following configuration will tell apt-get to load at least the English and German translations from the Debian server.

Acquire::Languages { "environment"; "en"; de"; };

If you don't want apt-get to load any languages and thus risk apt-cache to display any long descriptions use the following configuration. This is also the default setting if you upgrade an old Debian system to current distributions.

Acquire::Languages { "none"; };

**Brandnew Packages**

Packages uploaded into the Debian archive by a Debian developer will only be synchronized into the official archive once a day (this will be changed to twice per day in the future). Due to this limitation there may be a 30hour delay until a recently uploaded package hits the mirror you use.

Furthermore, the Debian project distinguishes between updated and new packages. Updated packages refer to updated versions of packages that already exist in the particular suite. They will be processed by the archive software automatically. New packages, however, do not yet exist in the particular suite and require manual attention.

An ftpmaster needs to inspect the new package and see if it is suitable for the Debian archive. If the origin is a source package which is already present in the archive only the internal database needs to be update to reflect the section and priority of the new package.

If the package originates in a source package which is also not yet present in the archive, more work needs to be done. An ftp master will have to inspect the package and decide whether it is suited for the Debian archive. They have to read the license carefully and reject the package if it does not conform to the [Debian Free Software Guidelines](http://www.debian.org/social_contract#guidelines).

Furthermore the Debian project is allowed to distribute cryptographic software. It is even allowed to export cryptographic software from the U.S. if it is Free Software. The U.S. government considers cryptographic software as weapons and hence usually do not permit to export it.

However, this requires the Debian project to [announce](http://www.debian.org/legal/cryptoinmain) all software that may be subject to cryptography to the U.S. Bureau of Export Administration (BXA). In order to comply to this regulation, all new packages are reported to the BXA with the note that they may contain or utilise cryptographic software.

This, unfortunately, requires the Debian project to not export such software before the BXA has been informed. Hence, the new queue of the incoming directory cannot be made public anymore. They will have to be kept private unless they have been properly processed, in which case they'll appear in the Debian archive at last one day later.

Packages sitting in the incoming directory will be added to the public package archive with the next archive software run. They can be fetched from the incoming [directory](http://incoming.debian.org/) directly when immediate updates may be required. This is a direct gateway to the accepted directory in which updated packages and processed new packages are stored until the archive maintenance run.

**Packages not in Debian**

The amount of Free Software is sheer endless. Today a lot of projects and individual produce very good applications and utilities that are distributed under a Free Software license. Naturally, not all such packages are part of the distribution, not even in the case of Debian which already refers to the largest archive of integrated packages.

If you are not using the unstable distribution, chances are that the developers have added the missing package to the archive already but it is simply not yet available in the stable or testing distribution. In that case it may be possible that you find a [backport](http://www.backports.org/) of the package.

When the distribution you are using is not too far away from the unstable distribution with regards to libraries and stuff, it may be possible that you can install the package from unstable. You'll find a download link via [packages.debian.org](http://packages.debian.org/). If this doesn't work, it may be possible to recompile the source on your system inad install the resulting package.

Please note that neither of this is recommended by the project and that all errors that may occur lay entirely in your responsibility. The package maintainer may be a source for help but they are not required to support their package on distributions it is not designed for.

A list of unofficial Debian archives is [www.apt-get.org](http://www.apt-get.org/). Packages from one of the listed sources are not supported by the Debian project but only by their respective maintainers. You may, however, find software that is not yet packaged for Debian because the integration is difficult or large parts of code are duplicated.

If you are looking for a particular package that is not even part of the unstable distribution and can't find it in the unofficial sources mentioned above, you may have success on [Fresmeat.net](http://freshmeat.net/) or [rpmfind.com](http://www.rpmfind.com/).

**Virtual packages**

A virtual package does not really exist in the package universe of Debian. It is a name other packages can depend on if they depend on a certain functionality. Virtual packages are used when several packages provide the same functionality that other packages need. The name of the virtual package often refers to the functionality and not to a common package name.

Instead of declaring a dependency against five distinct packages it is sufficient to only declare a single dependency against the virtual package which is provided by all of these five packages.

In order to help the dependency resolver often a dependency is declared on the virtual package and one of the packages providing it. This way the resolver will prefer one particular package if none of them is installed yet.

A common example is mail-transport-agent which is provided by Postfix, Sendmail, Smail, Exim, Nullmailer, ssmtp and other packages that provide /usr/sbin/sendmail as a means to accept and deliver mail.

The list of virtual packages is documented on the Debian [website](http://www.debian.org/doc/packaging-manuals/virtual-package-names-list.txt).

**Pseudo packages**

Unlike virtual packages pseudo packages really don't exist. They are used so that people can assign but reports to certain parts of the distribution or the Debian project that doesn't originate in a source package. Hence, they don't refer to a source or binary package.

This way it is possible to assign bug reports to www.debian.org for example. The entire website does not originate in a particular package but is maintainer by webmasters who are improving the website and fixing bugs as well. Instead of contacting them directly, users may rather open a bug report against this pseudo package.

Another example would be the installation-reports pseudo package which is used to collect reports of new installations. These are used by the [debian-installer](http://www.debian.org/devel/debian-installer/) team to check with new installations and discover bugs that need to be fixed.

The list of pseudo packages is documented on the Debian [website](http://www.debian.org/Bugs/pseudo-packages).

**Package Pools and Distributions**

The Debian archive is organized in distributions (stable, testing and unstable). They refer to a directory below the dists directory in the archive top. In these directories the index files for source and binary packages of the distributions reside.

Starting with sarge (Debian GNU/Linux 3.1) the package files itself are stored in a different directory. Past distributions kept the source and binary packages within these directories as well. Today they only contain index files.

Instead the packages itself are stored in the pool directory in the archive top directory. All current versions and architectures are stored in the same directory. The index files mentioned above refer to files within this directory, so that APT is still able to download packages.

In the pool directory the package files are stored in a special directory named after the source package. This directory is placed in a directory consisting of the first letter (or lib in the case of libraries) in order to reduce the number of sub-directories and to reduce the delay when reading the directory.

**The testing Distribution**

This distribution has been introduced in order to help prepare the next stable distribution. The testing distribution should be releasable all the time. Packages in this distribution should work together fine and all dependencies should be resolved. Additionally, all architectures should always be in sync.

Packages uploaded into the unstable distribution will migrate into the testing distribution automatically if they have been proven stable for a number of days, all architectures are in sync and all dependencies are fulfilled in the testing distribution itself.

The quarantine time for a package with urgency low is 10 days, with urgency medium 5 days and with urgency high two days. No [release critical](http://bugs.debian.org/release-critical/) bug must be reported during this time.

This distribution should always be in a state in which it could be frozen and declared stable. Packages in this distribution work sufficiently well and don't incorporate the problems of the unstable distribution. However, there is no security support available. Security updates will migrate into testing the same way as normal packages, however, they usually declare an urgency of high.

**Forcing Package Installation**

When you try to install packages from unstable on a stable distribution or try to install packages that conflict, dpkg will spit out error messages refusing to install these packages. This is good, since it will keep the user from destroying his system.

However, for advanced users or developers there are situations where dpkg is wrong and the package should be installed anyway. In that case, please call dpkg --force-help to find out about its interface to force things. The following is only a short list with some explanation, dpkg is capable of more.

* --force-overwrite forces dpkg to overwrite files that are already installed by a different packages.
* --force-depends forces dpkg to install or remove packages even if this causes some dependencies not to be fulfilled.
* --force-conflicts forces dpkg to install packages even if they conflict.

These parameters may only be used in rare exceptional cases and must not be used as default setting since they will harm the system in most cases. Only advanced users may use them if they know how the system behaves and what their use will result in.

**Creating Local Archives**

This section is most probably only useful for experienced users who need to maintain additional resources for apt. This may be useful if you want to provide a repository containing current versions of a software product that is not (yet) part of Debian but consist of more than one binary package (e.g. KDE, OpenOffice, GNOME etc.) or if you want to use Debian for your corporate network and need a locally maintained archive for updates.

The programs to generate your Packages and Sources files are dpkg-scanpackages(8) and dpkg-scansources(8) respectively. Both are included in the dpkg-dev package.

Whenever you consider providing your own archive please read the manpage for dpkg-scanpackages(8) carefully. Lets assume that you wish to provide an archive that will be covered by the following line for sources.list

deb http://debian.domain.org/debian unstable foo

and DocumentRoot of your webserver is /var/www/, consider the following directory structure

/var/www/debian

../dists

../unstable

../foo/binary-arch/

../foo/binary-all/

../foo/source/

Where arch is a valid architecture string (like i386, ia32, m68k etc.). By comparing the line for apt-get with the directory structure you should easily be able to support other architectures, other distributions and other sections by simply adding more directory trees to the proper root.

Both programs, dpkg-scanpackages and dpkg-scansources, will descend into subdirectories and handle them properly. That's the way the main debian archive works by using subsections (like mail, base etc.). Both programs will also follow symlinks which means that binary-all programs may indeed be symlinks to ../binary-all/program.deb. This feature is also used on the main debian archive.

In order to generate the Packages file you have to issue the following two commands. It is important to change the directory to the main Debian directory first so all paths will be adjusted properly.

cd /var/www/debian

dpkg-scanpackages dists/unstable/foo/binary-arch /dev/null \

> dists/unstable/foo/binary-arch/Packages

The component /dev/null could be an override file (which you could fetch from [debian.org](ftp://ftp.debian.org/debian/indices/)) you probably don't have, thus it is ignored that way.

The above paragraphs discussed the setup for the large archives which is used at debian.org as well. If you only plan to provide a small archive and only support one architecture you can probably skip large chunks of the above. Assume you want to provide the following line for sources.list:

deb http://debian.domain.org/debian foo/

If you don't forget to write the trailing slash, apt-get will expect the package file in the directory /var/www/debian/foo. As a result of this, you can place all .deb files in that directory directly without having to care about general directory structures. Creating the Packages file is easier as well:

cd /var/www/debian

dpkg-scanpackages foo /dev/null > foo/Packages

Please be warned that these scan programs may not handle multiple versions of packges properly. Thus, if there are two versions of the same package it may be possible that the newer version doesn't appear in the Packages files but only older ones. You'd better try to avoid that situation.

**Upgrading single Packages**

To upgrade a single package, fetch the package and install it with dpkg -i file.deb or perform both steps with apt-get install file, however this requires a properly set up sources.list file.

When you want to install files from testing or unstable on a system that reflects the stable distribution of Debian, you may run into dependency problems. This will happen especially if you are trying to update some major or large package (like sql, xfree, gnome, kde etc.). They will probably require a newer glibc, newer xlibs etc. Due to this a regular upgrade of everything may be a good idea.

**Upgrading everything**

To upgrade from one distribution (e.g. stable) to a newer revision of the same distribution you only need to issue two commands. This is needed when you want to incorporate all security updates from your last upgrade or your first installation, or when the Debian project has released a new revision of the stable distribution. Your sources.list (see [APT Sources, Section 3.8](http://www.infodrom.org/Debian/doc/maint/Maintenance-pkgmaint.html#s-apt-sources) for details) would look like this:

deb http://ftp.debian.org/debian stable main

deb http://security.debian.org stable/updates main

If you also want to use non-free parts that were packaged for Debian, you'll have to add contrib non-free after the word main. After that the following commands will do everything needed.

apt-get update

apt-get upgrade

The first command will fetch new package databases from the archive and the latter will actually fetch packages and install them. If you only want to fetch them but not install them please add the -d switch to the second command line. If your Internet connection goes down during the update, re-issue the command and apt-get will restart the process.

**Distribution-Upgrade**

Every once in a while, even the Debian project releases a new stable distribution. These releases usually have accumulated quite a number of new and updated packages. Many libraries are available in a newer version and security support for the old stable release will end in one year after the new release.

Because many libraries have been modified and several new libraries have been introduced, simply upgrading every single package doesn't work. Instead the package manager needs to calculate all dependencies and decide which package to update and which new ones to install. The installed packages also need to be grouped in small sets which also need to be calculated.

As a general rule you should always check the [release notes](http://www.debian.org/releases/stable/releasenotes) before starting the upgrade. You should also backup all of your configuration and all-important data that may suffer from the upgrade. It will mention those packages that need to be updated before the rest of the system when problems are anticipated otherwise.

If you have enough disk space on one of your systems, you could also test the upgrade before actually performing it. The package [dchroot](http://packages.debian.org/dchroot) allows you to switch into a chroot environment. You could clone your production system into another directory, jump into it and perform the distribution upgrade inside the cloned system. Without risking your production system, you could test the upgrade and develop solutions to problems in case you should experience some.

The same applies to you if you wish to upgrade from a stable or testing distribution to unstable.

**Upgrade with apt-get**

A distribution upgrade usually works with apt-get dist-upgrade after the package database has been updated. However, due to the complexity of the Debian distribution it is possible that APT can't cope with the dependencies as it should and will offer to remove several packages.

When this happens, it is often helpful to manually upgrade certain parts of the distribution. This is done by executing apt-get install on some of the packages that should be removed otherwise. This will ensure that they are upgraded together with their dependencies.

Candidates for such particular upgrades are dpkg, apt, libc6, apache, several libraries, several GNOME and KDE packages. Afterwards a regular dist-upgrade should work again.

**Upgrade with aptitude**

When you choose to upgrade your distribution via [aptitude](http://packages.debian.org/aptitude) you should upgrade this package manager first, either with apt-get or with aptitude itself.

Afterwards the regular upgrade of the entire system should work flawlessly. Execute aptitude -f --with-recommends dist-upgrade to perform the upgrade. If too many packages are to be removed, please upgrade some packages manually just as with apt-get.