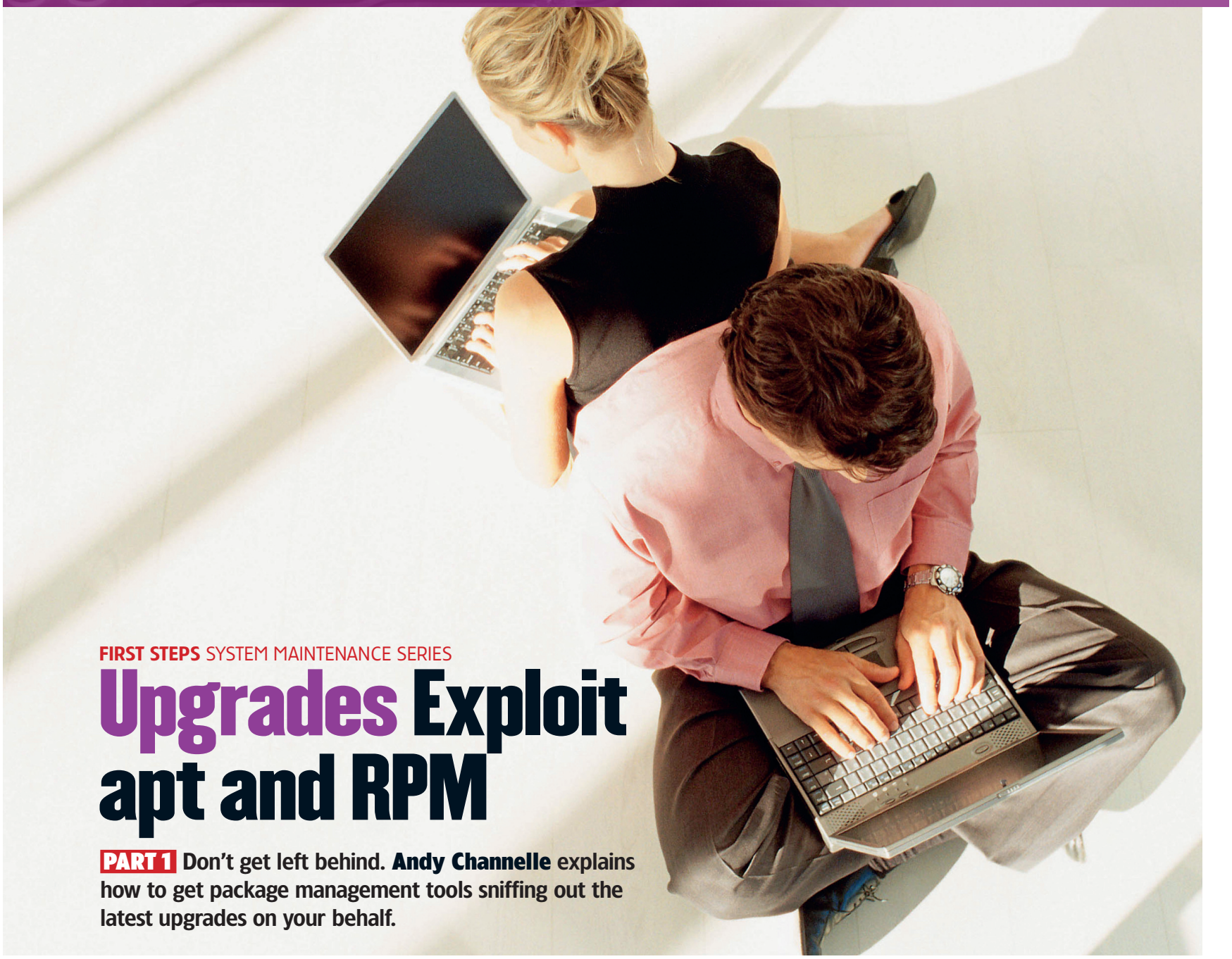


FIRST STEPS SYSTEM MAINTENANCE SERIES

# Upgrades Exploit apt and RPM

**PART 1 Don't get left behind. Andy Channelle explains how to get package management tools sniffing out the latest upgrades on your behalf.**

There is a bizarre disconnection between people's perception of using Linux and the actual experience of maintaining a system. It is commonly believed – though this is slowly changing – that software is hard to install; that there are dependencies that will remain forever unsatisfied; and that installing one thing breaks something else.

The reality is very different. Imagine, for example, having a single application on a Windows XP or Macintosh system that could download and install any bit of software you might ever need – for free. Moreover, this piece of wizardry would monitor applications on the host system and on a remote server and then, should an update become available, inform you of the change and offer to perform the upgrade with very little interruption for you.

**"IMAGINE HAVING A SINGLE APP THAT COULD INSTALL ANY BIT OF SOFTWARE YOU MIGHT NEED."**

It would be magic! Revolutionary! Impossible!

Except that it *is* possible – almost every Linux distribution available today has just such a system installed; what's more, they work. And things are likely to only get better, as distribution vendors compete with each other to offer the simplest method of system management.

This tutorial will look at the process of maintaining *DEB*- and *RPM*-based distributions. Examples of the former include Debian, Ubuntu, Xandros and Linspire; the latter group includes Red Hat (which gave the technology its name: *Red Hat Package Manager*), Mandriva and SUSE. Though there are distribution-specific options for maintaining your system, among them *YaST*, *Xandros Networks* and Click-N-Run, I will be concentrating on the more generic tools. We'll start with Debian's *Advanced Package Tool* (*apt*), then move on to *RPM* before looking at *apt4rpm*, which brings the two main systems together.

All of this will involve the use of the command line, but it's worth noting that tools such as *Synaptic* and its KDE equivalent, *Kynaptic*, bring many of these options into the graphical realm if you find it easier to work that way. Even so, an understanding of what's going on in the background will make for a more satisfying experience.

## OPTION 1 – UPGRADE WITH APT

***Apt* was designed as a tool for installing and updating** software via a network, and is ideal for users who have a decent – ideally broadband – connection to the internet. The system uses shared repositories, which can be local, networked or online. Within the Debian community there are both official and unofficial repositories; and to confuse users even more, official Debian repositories can belong to either the 'stable', 'unstable' or 'testing' branches. The most up-to-date options, the sort of software that project leaders fear could damage the distribution's reputation for stability, reside in the testing branch, while the tried and tested stuff is in stable.

On a basic Debian system, telling the OS where it can get software from is not particularly intuitive: it involves editing the file **sources.list** in the directory **/etc/apt/**, for which you will need to be the root user. This file will contain a number of lines, and the most important ones will use this formula:

```
deb http://hostname/distribution section_1 section_2
```

This line breaks down into various elements. The first (**deb**) tells us that the repository holds binary files – that is, pre-compiled applications that can be installed as they are. If **deb** has an **-src** suffix it means that the repository contains source files complete with a Debian control file and an archive detailing the extras needed to create an installable binary.

The second element is the repository's URL. Debian can deal with HTTP (which is what the world wide web runs on), FTP, file (for local sources) and SSH, for more secure communications. This bit follows the usual URL conventions. The *Scribus* Debian repository, for example, is available at **www.scribus.org.uk/downloads/1.2.1/debian binary.**

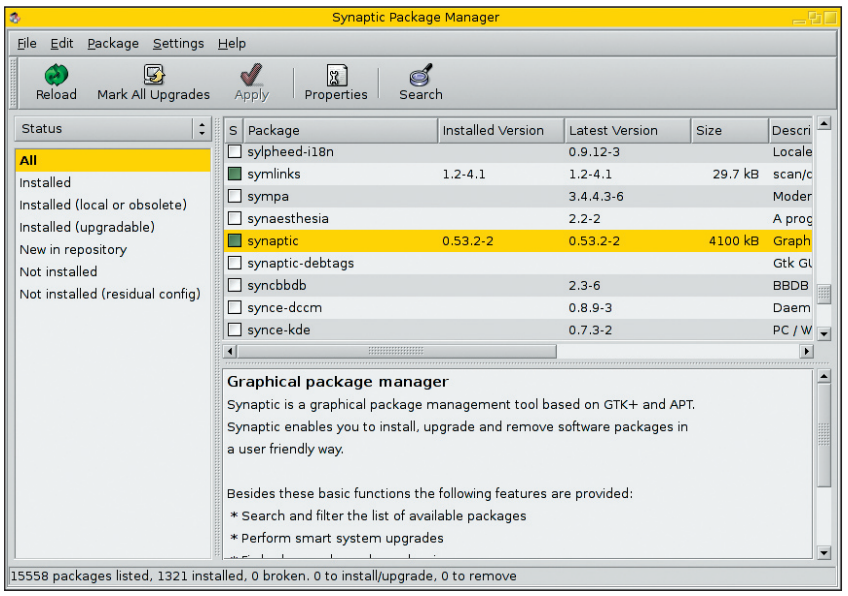
The final elements in the line are optional and direct the package management software to different sections of the repository. Normal Debian repositories will have a section dedicated to the branch (whether the upgrade is in stable, unstable or testing) and also to sections called 'main', 'contrib' and 'non-free'.

A normal **sources.list** file might look like this:

```
# Remember that you can only use HTTP, FTP, SSH or file URLs
# CDROMs are managed through the apt-cdrom tool.
deb http://http.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable/non-US main contrib non-free
deb http://security.debian.org stable/updates main contrib non-free
#Uncomment if you want the apt-get source function to work
#deb-src http://http.us.debian.org/debian stable main contrib non-free
#deb-src http://non-us.debian.org/debian-non-US stable/non-US main contrib non-free
```

### FEDORA YUM

**Fedora users have their own native *apt*-like system called *yum* (Yellow Dog Update Manager). It uses a similar syntax to *apt*, but here are a couple of tips all the same. Before using the system for the first time, run yum check-update to download information about the applications from the remote repository. You can install software with yum install [package], and remove with yum remove [package]. Finally, all available updates can be installed by running yum update. As you're changing your system here, these commands will need to be run as root.**




**Installing *Synaptic* (apt-get install synaptic) will give you a lovely front-end for Debian's package manager.**

Lines with a hash sign in front of them are not read by the system This means that we could add support for source repositories from the list above by simply removing that character from the last two lines. Distributions from Ubuntu, Xandros and Linspire use specific repositories with software specially packaged for their environments, and while it's possible to use a wide range of Debian repositories, please be aware that by going out of the 'walled garden' you risk messing up some pre-installed piece(s) of software. Back up your data and system files before you make any substantial changes!

One thing to remember is that whenever you change something within this file, you should follow it up by opening the console and typing

```
apt-get update
```

This ensures that the system has the most up-to-date information on what's available from the repositories. On a slow connection, this could take some time. A full list of complete Debian mirrors is available at **www.debian.org/mirror/mirrors_full**. There are several in the UK, which should provide the best speed for British users.

If you have a full set of Debian CDs, you can add these to **sources.list** from the command line. Insert the CD and type

```
apt-cdrom add
```

If the disc drive mounts to some unusual location, add a **-d** switch and the path to the device.

```
apt-cdrom -d /media/cd0 add
```

As well as official Debian sources, there are unofficial software repositories such as the *Scribus* one mentioned above. These usually provide an address which can be cut and pasted straight into **/etc/apt/sources.list** for access to the software. Once done, do **apt-get update**.

### Installing software

Despite its reputation for being difficult, there is one big reason why Debian is used as a base for all the user-friendly distributions such as Linspire and Lycoris – it has brilliant package management tools. For a very easy example of this, open a console/terminal on any Debian-based system and, as root, type

```
apt-get install gimp
```

### QUICK TIP

**When to run as root**
**If running as root involves typing commands in a console, the way to 'become root' is to type su then your root password. You can then issue commands as though you were the system administrator. Another application of root commands: if you needed to edit sources.list, for example, you could become root at the console and open a text editor by typing kwrite or gedit (in KDE and Gnome respectively) to start the app with root privileges.**

This should download the latest version of the *Gimp* image manipulation application available from the locations defined in **sources.list** and add it to a sensible place on your start panel.

Debian package maintainers also prime their packages with a list of dependencies so that *apt* can scour the repositories for those too, so there should be no scratching around for some obscure library file. You can also remove applications using the same method, thus:

```
apt-get remove gimp
```

While it is working, the software will tell you precisely what it intends to do – eg 'upgrade three files, install two and remove one' – and will ask for your assent.

```
Do you want to continue? [Y/N]
```

Hit Y to continue.

Debian repositories contain thousands of applications, so you can be pretty sure that whatever you're looking for is out there somewhere. If you know what you want but are not sure of its name, *apt* allows you to search the repositories using the command *apt-cache search*. We might, for example, want to play the newly updated open source *Civilization*-style game whose name escapes us, so we could try

```
apt-cache search civilization
```

which would give a list of packages that have 'civilization' (or part of it) in their name or description – including one called *Freeciv*. Now, I think that's the right one, but if I want to be extra sure, I can do

```
apt-cache show freeciv
```

to look at a more detailed description of the application. Now that I'm sure this is what I'm looking for, I install with **apt-get install freeciv** and start playing.

Upgrading applications is just as simple as installing them. Running the command

```
apt-get upgrade
```

will examine the contents of your system, compare it with what's in the repository and offer to update local versions that have changed. Always make sure that you run **apt-get update** before doing this so that *apt* is aware of the most recent versions available. Again, the system will give you information about how many applications are going to be altered, added or removed: you should also see a list of everything you can upgrade, and you'll be offered the chance to accept or decline the upgrade.

As if that weren't simple enough, it is also possible to use *apt* to effect a system-wide update. For example, users of Ubuntu Warty Warthog or Debian 2.0 should, in theory, be able to open a terminal, do **apt-get update** then issue the following equally simple command

```
apt-get dist upgrade
```

and sit back with a nice cup of tea as the entire distribution is upgraded to the very latest version. Which is very cool. Back up though, just in case. You'll get no sympathy from anyone if you lose all your data and don't even have a copy to hand.

## OPTION 2 – UPGRADE WITH RPM

**Where Debian has *apt*, the likes of Red Hat, SUSE and** Mandriva have *RPM*. This is a package management system, but the term RPM is also used for individual packages, such as the KDE RPMs. Oh, and it's a command, too. It doesn't offer the dependency management of *apt*, but still does a good job as long as you stick to using only RPMs designed for your specific distribution and version number. It's possible, for example, to use RPMs meant for SUSE 9.1 on a SUSE 9.3 system (and sometimes even Red Hat binaries on Mandriva), but the potential for breaking something increases the further away you get from your actual distro.

For the purposes of this tutorial I have on my hard disk a fictional application called *andybob-2.rpm*, and I install it by typing the following in a console with root privileges:

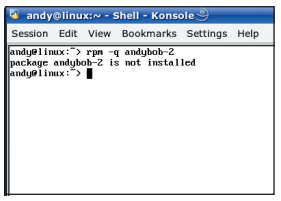
```
rpm -ivh andybob-2.rpm
```

This is a short command, but it does quite a lot. The initial **rpm** invokes the actual package management software, which in turn is modified by the arguments (**-ivh**). These are passed on to the package name. (It's quite possible, by the way, to fill a directory with RPM files and then install them all using a wild card for the package name, eg *.rpm). The arguments used above will install the file (**i**), provide verbose output so you can see what's going on (**v**) and put a line of hash symbols (**h**) across the screen as a sort of progress bar. If you don't need this sort of feedback, just use **rpm -i andybob-2.rpm**.

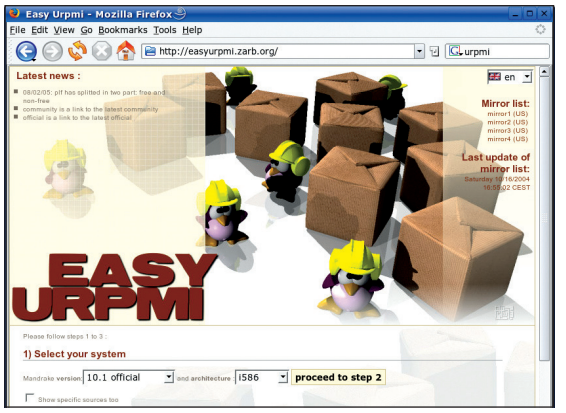
It's very unusual to see advanced Linuxers using *rpm -i*, because *rpm -U* works so much better. In the above example, *RPM* would simply install *andybob-2* regardless of what is already available on your disk – even if that includes *andybob1*. Using the **-U** switch means that *RPM* will install the application if no previous version exists – but if you have *andybob-1* it will simply update it, and save you the confusion of having two versions of the same application on one PC.

*RPM* can remove software just as easily as *apt* – for this we use the **-e** or erase switch.

```
rpm -e andybob-2
```




***RPM* tells us that *Andybob2* is definitely not installed on this computer.**




***Easy URPMI* (http://easyurpmi.zarb.org) is a great tool for Mandriva/Mandrake users who want to update their systems remotely.**

This uninstall process is silent – that is, it will not print any data to screen as it is working – but once it is complete, the cursor will return to the console.

*RPM* can do more than just install and remove software though. Imagine for a moment that someone mentions a great application called *andybob-2* and recommends installing it. You think you may already have this, but to check, open the console and type

```
rpm -q andybob-2
```

The **q** switch simply means 'query.' If the application is not installed, the console should print out 'andybob-2 is not installed.' If it is there, the name and version number will appear on the next line. If you have this groundbreaking application installed, but are unsure of the many fictional benefits of using it, you could get more information using **-qpi**, which will print information such as the version number, group (games, internet, multimedia, etc), size, creation date and, crucially, a description of what it actually does.

## OPTION 3 – DIY WITH APT4RPM OR SOURCE CODE

**Having two competing formats is good for users, so the** thinking goes, because they end up with better features and efficiency thanks to the nature of competition. This is true in the Linux world: we seem to have two of everything. But it is also in the nature of Linux that if there is two of something, some clever so-and-so will try to make them work together.

And so it is with the main competing package management systems. *RPM* and *apt* were merged by engineers at the company formerly known as Conectiva (itself now merged with Mandrake, of course) into a system called *apt4rpm*.

The problem with this is that each distribution with a version of *apt4rpm* available must also maintain an *apt* repository. There are versions of *apt4rpm* available for Red Hat/Fedora, SUSE, Yellow Dog and Mandriva, and a list of repositories is available from the project's SourceForge site at **http://apt4rpm.sourceforge.net**.

I am in the process of adding this facility to my SUSE desktop, and the files I need are available from **ftp://ftp.gwdg.de/pub/linux/suse/apt/SuSE/9.2-i386/RPMS.suser-rbos**. *RPM* can work with fully qualified IP addresses or URLs, so I can install the application from the console with one of two commands. Beware the two opposing slashes you'll see after **suser-rbos**; these tell *RPM* which files are needed from the directory. Remember this all needs to be done as root, so in the console you can first type **su** and, when requested, your root password. The first things we need are the *apt* libraries:

```
rpm -ivh ftp://ftp.gwdg.de/pub/linux/suse/apt/SuSE/9.2-i386/
RPMS.suser-rbos/\apt-libs-0.5.15cnc6-rb.suse092.6.i586.rpm
```

Then we need *apt* itself:

```
rpm -ivh ftp://ftp.gwdg.de/pub/linux/suse/apt/SuSE/9.2-i386/
RPMS.suser-rbos/\apt-0.5.15cnc6-rb.suse092.6.i586.rpm
```

These are meant for SUSE 9.2, but I had no problem installing them on the latest 9.3 release. Once this has finished (you could do **rpm -q apt** to check the installation went smoothly) the last job is to add a repository to **sources.list** which, as with an ordinary Debian installation, will be in **/etc/apt**. Open this file and add the line:

```
rpm ftp://ftp.gwdg.de/pub/linux/suse/apt/SuSE/9.2-i386 base
update security
```

Please note that, though the *apt4rpm* installation for SUSE 9.2 worked OK on SUSE 9.3, any attempt to run **apt-get upgrade** would be insane, because the repository contains applications for 9.2 that have been superseded in the latest release. Ensure that you have a working repository that corresponds exactly to the distro you are using. And back up your data and/or system before making any significant changes.

Fedora Core 3 users can install *apt4rpm* with

```
rpm -ivh http://ftp.freshrpms.net/pub/freshrpms/fedora/
linux/3/apt/\apt-0.5.15cnc6-1.1.fc3.fr.i386.rpm
```

and repositories for **sources.list** are

```
rpm http://ayo.freshrpms.net fedora/linux/3/i386 core updates
freshrpms
rpm http://ayo.freshrpms.net fedora/linux/3/i386 tupdates
```

Mandrake 10.1 users, meanwhile, can do

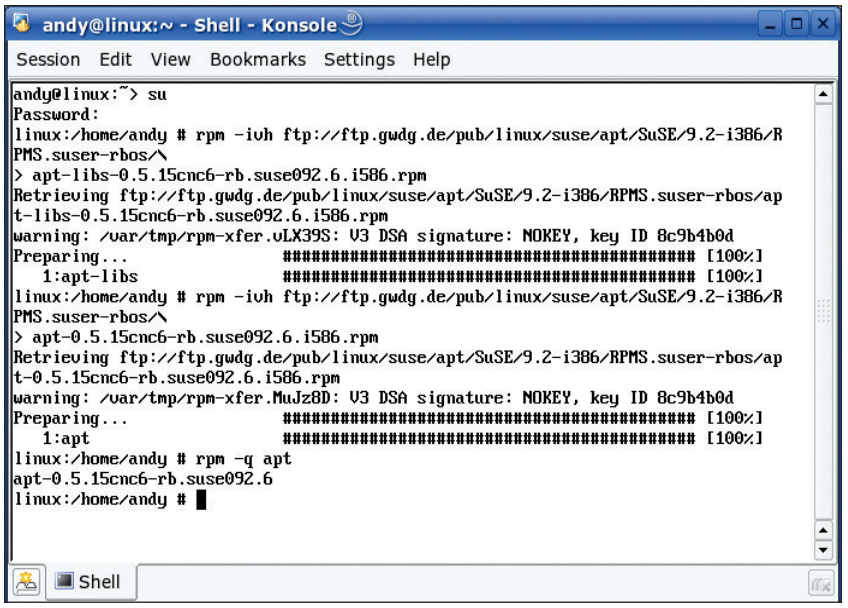
```
rpm -ivh ftp://fr2.rpmfind.net/linux/Mandrake/10.1/i586/media/
contrib/\libapt-pkg0-0.5.15cnc6-3mdk.i586.rpm
rpm -ivh ftp://fr2.rpmfind.net/linux/Mandrake/10.1/i586/media/
contrib/\apt-0.5.15cnc6-3mdk.i586.rpm
```

and add the following to their **sources.list**:

```
rpm ftp://ftp.lip6.fr/pub/linux/distributions/Mandrakelinux/
official/10.1/i586/media/ media_info/hdlist_main main
```

If you like a little adventure it is possible to download source code for applications and compile it yourself. The advantages of this method include gaining access to the most current features and having applications optimised for your particular system. Bad points include the potential for falling into dependency hell, where application X needs library Y, while library Y needs library Z and library Z needs a sample of DNA from your first-born. If you're compiling development stuff, there's also the possibility that you'll end up with software riddled with bugs and incomplete features. Oh, and you'll also need a ton of development tools from your distribution's CD-ROM, probably including the kernel source. And compiling can take a loooong time.

Still with me? Good.




***RPM* installing the SUSE version of *apt4rpm*.**

> **"THE ADVANTAGES INCLUDE HAVING APPS OPTIMISED FOR YOUR PARTICULAR SYSTEM."**

Source code tends to come in an archive, usually with the file extension .tar.gz. Most software will tell you that, once downloaded, you can extract the software with

```
tar –xvzf ./andybob-1.0.tar.gz
```

You can get the same effect by right-clicking on the file in *Konqueror* and selecting Extract To Here... Next, you navigate to wherever the extracted files with

```
cd /home/andybob
```

And run the configuration script:

```
./configure
```

Once this is complete you run the **make** command to build the binary before concluding the installation itself with **make install**. If you are in any way worried about the final part of this process, you can add **test** to the **make install** command to simulate the installation without touching the system.

Linux really does make it easy to keep your system up to date, often informing you of upgrades without even being prompted. Now if you'll excuse me, I must go and upgrade the latest version of *Kino* – just so I'm ready for that call from Spielberg, you understand. LXF

**NEXT MONTH**

**We'll be looking at new features available in KDE 3.4; including *Akregator*, the new integrated tool for monitoring RSS news sources.**