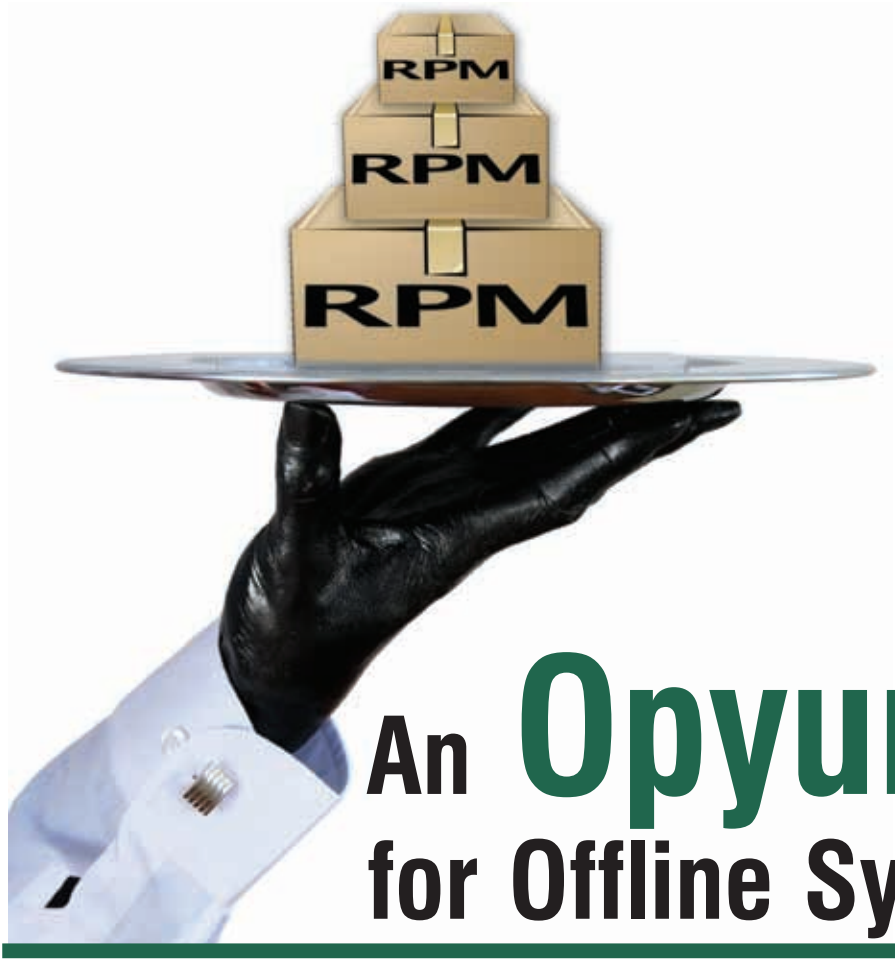

A DVD holds a lot of software. But the problem is how to keep your OS current with the latest mandatory updates or install additional software when you're hampered by a slow Internet connection. Introducing Opyum, an offline software installer/updater for Fedora. And if you want it on your Debian too, why not help port it?

An Opyum Drag for Offline System Update

Opyum (pronounced 'opium') is a package management tool meant for those who do not have cheap and fast access to the Fedora repositories. Most often, this is due to the absence of broadband Internet connections, but one can imagine other scenarios as well. The name expands to 'Offline Package Management using Yum', and provides an interface that is very similar to that of Pirut, Fedora's 'Add/Remove Software' utility.

These days, every mainstream GNU/Linux distribution provides a comprehensive set of programs for users to install on their system. Many of these programs are regularly updated and made available, too. Powerful tools like Apt, Yum, etc, make it very easy to delete, install or update these programs.

However, to avail this facility, you must have a cheap and fast Internet connection. Without it, updates are out of the question, and you become restricted to the smaller subset of packages that are present in the installation DVD. Unfortunately, in countries where Internet bandwidth is very costly (like in India), a considerable chunk of existing or potential users face this problem.

Opyum tries to provide a solution. Ubuntu already has something similar written by Baishampayan Ghose during last year's Summer of Code, so this time I chose to focus on Fedora.

How does it work?

The basic idea is to download the necessary packages required for an installation or update on a different machine that has access to the repositories. Once done, use some removable offline media to transfer them to the site, where the actual installation or update needs to take place. Deletion is not an issue, since it is inherently an offline activity.

Imagine you have access to two different systems—one at your home without a network connection, and one at your workplace with Internet access. So you do the download at work, bring home the packages on a pen drive, and do the installation on your home computer. Sounds good enough, doesn't it?

Easier said than done though, because your office computer would not know about the profile of your home system. For example, if you want to install a new media player at home, you would need a bunch of plug-ins and codecs for the player to work. Depending on whether your home system already has these plug-ins and codecs installed

DEBARSHI ON HIMSELF AND HIS SOC PROJECT

Born and brought up in Kolkata, I passed out from National Institute of Technology, Hamirpur, this year as a computer science and engineering graduate. My first brush with GNU/Linux was a long time ago, when my father brought home a Red Hat Linux 6.2 boxed set. It was bought by his office, but since nobody was interested in it, it found its place in a corner of my bedroom. After a few unsuccessful attempts to install it, my Free Software aspirations lay in cold storage for the next five years until I was in college.

Times had changed, I had a laptop now, and it was refreshingly easy to get Fedora Core 1 to run on it.

Right from my childhood, I have always been fascinated by the Help → About dialogue of an application—the strange looking version string, the weird language of the copyright and licence notices, and the names of the individuals or companies associated with the application and their respective e-mail addresses. I often imagined how it would be to have a microsoft.com or adobe.com e-mail address. Gradually, this changed to sun.com, then to redhat.com and then to gnu.org, gnome.org, fedoraproject.org, etc. Yes, the code was interesting, but the e-mail address and having your name in the credits list was even more so. And the best thing about Free Software programs was that most of them had the names of the actual developers on the Help → About dialogue compared to their proprietary counterparts.

However, when the first edition of Google Summer of Code was announced in 2005, I was lost. All the big names that I had come across on my computer were there as mentoring organisations, but I could not make any sense of the suggested proposals on the website. There was this one which said something about improving GNOME's start-up time. That sounded fantastic, but how on earth would this be possible? It looked like black magic to me.

The same thing happened in 2006.

But this year was different. A few good things had happened in between 2006 and 2007. My friend Rakesh and I did a summer internship under Dr Nagarjuna at the TIFR, which was kind of ground-breaking; and I had gained some knowledge too. The black magic was less black now, and since I was in my final year, I had to make it this time. Otherwise, a good-looking e-mail ID could just remain a distant dream forever!

The problem of not being able to use package management tools like Yum and Apt because of the lack of an Internet connection was all around me. Baishampayan Ghose had already been selected in 2006 for trying to solve the problem for Ubuntu. So it looked 'safe' to submit a similar proposal for Fedora this time. After some discussions with Sankarshan Mukhopadhyay, who became my mentor, and Rahul Sundaram, I submitted it.

It got selected, and the journey began.

or not, you would need to download them in addition to the media player at your office. The issue gets more complicated if you are trying to update your entire system, where you would need to know a few thousand packages.

Every Fedora system has a small database residing in /var/lib/rpm that contains detailed information about the current set of packages installed in the computer—the profile. This database is used by all existing package management tools like Pirut, Yum, or even the command-line utility *rpm*. So if you could carry the profile of your home computer to your workplace and use the information to decide about the downloads, your problem is solved.

This adds an extra step to the whole process, which can be broken up into the following steps:

- (At home) Start Opyum and export your profile on any removable offline media, viz. a pen drive.
- (At office) Import the profile and select the packages you wish to install or update using Opyum. Once the downloading is complete, the packages will be saved in a Yum-Pack, which can be placed on any removable offline media like you did with the profile.
- (At home) Double-click the Yum-Pack to complete the transaction.

Only the last step would require root permission on the

home system.

Possible scenarios

Apart from the basic objective of doing offline package management, you can think of a number of other interesting scenarios where Opyum might be useful.

Newbies (possibly your parents and grandparents) are often overwhelmed by the huge list of available packages in the repositories, and are at a loss to figure out which program is going to do the job for them. The weird names do not help matters either. Since exporting profiles and using Yum-Packs are both 'single-click' activities, you can ask them to provide you with a copy of their profiles so that you could create a Yum-Pack containing all the packages that would meet their needs. Instead of staring at a list of more than 8,000 packages, which makes no sense to them, they would only need to click the Yum-Pack, provide the root password and wait for the installation to be over.

Magazines providing the latest GNU/Linux DVDs can also provide Yum-Packs containing restricted codecs and drivers that can not be officially a part of those distributions because they are based in countries that recognise software patents. Distributions, like Fedora, would not allow you to provide such restricted packages on the installation DVD, unless you completely remove the Fedora branding. Yum-Packs based on a very minimal profile is a possible solution that can be provided to users, who would not need to hunt for essential codecs and drivers all over the Internet.

Looking under the hood

Internally, Opyum uses the Pirut back-end to do most of the work. Pirut provides the bulk of the GUI components and, in turn, uses the Yum API to do all the dirty work involving packages. Only those portions that are specific to Opyum—profile management, creation and use of Yum-Packs—are kept outside the Pirut core.

Following is a simplified view of the main PackageManager class. It inherits the Pirut back-end from the GraphicalYumBase class provided by Pirut.

```
class PackageManager(GraphicalYumBase):
    """
    The basic package manager class to do the back-end work.
    """
    ...
    def findProfiles (self):
    ...
    def _apply(self, *args):
    ...
    def _exportProfile(self, widget, event=None):
    ...
    def _importProfile(self, widget, event=None):
    ...
    def _selectProfile(self, widget, path, column=None):
    ...
    def createYumPack(self, yumpack):
    ...
```

The main Opyum program does everything except using Yum-Packs to install or update packages on the standalone system. That is done by a smaller program called *system-install-yumpacks*, which is a part of the *opyum-0.0.3* package in Fedora. It is a smaller program, which inherits the Pirut back-end like the main application, but restricts itself to the task of installing and updating packages from the contents of a Yum-Pack.

```
class YumPackInstaller(GraphicalYumBase):
    ...
    def extractYumPacks(self):
    ...
    def populatePackages(self):
    ...
    def _apply(self, *args):
    ...
```

The choice of language is Python, because both Pirut and Yum are written using it, and for the same reasons PyGtk is used to provide the interface. However, much of this is going to change because the idea is to re-write Opyum using the PackageKit API.

PackageKit provides a DBus-level abstraction, which can be used to write cross-distribution package management tools. The CLI or GUI interfaces would no longer need to target a specific package management system, since the distribution-specific back-ends like Yum/RPM and Apt/Deb would be handled by PackageKit itself. As of now, the PackageKit's Yum back-end is already done, and work is underway on the Apt back-end. Similarly, there is work on to develop a bunch of distribution-independent command-line, Gtk+ and Qt front-ends.

The benefits are obvious, and once Opyum is migrated to PackageKit, it would no longer be specific to Fedora and its derivatives. Creating a PyQt alternative would also be easier.

Wrapping up

Opyum 0.0.3 (opyum-0.0.3) is available for Fedora 8, and would be soon available as an update for Fedora 7. You can read about it on https://fedoraproject.org/wiki/DebarshiRay/Opyum

Bugs (if any) are to be filed in the Red Hat Bugzilla (https://bugzilla.redhat.com/) against the component named '*opyum*'. You can catch me in #fedora or #fedora-india on irc.freenode.net, where I hang out as 'rishi' or 'debarshi'.

If you want to participate in developing Opyum, drop an e-mail at fedora-devel-list@redhat.com and have a look at https://hosted.fedoraproject.org/projects/opyum/ END

By: Debarshi 'Rishi' Ray