MANAGING SOFTWARE WITH 'urpm'

Mandriva's `urpm` ('User RPM') tackles several weaknesses of the lower-level `rpm` tool. It provides a system of automatically solving dependencies by offering to install or uninstall dependent packages. RPM only tells you which files are missing or which packages would be broken. It stores a full dependency set of any given location. RPM only stores data of locally installed packages. If configured, it fetches packages from the network upon installation, if they are newer than the packages on the installation media. It also allows easier and more extensive package queries, automated updates and more.

`urpm` isn't meant as a replacement for RPM, it is meant to make common RPM tasks easier. In many ways it resembles the package tool the of the Debian GNU/Linux distribution, ***APT***.

Configuring `urpm`

The central configuration file is `/etc/urpmi/urpmi.cfg`, the flat text data files are located in`/var/lib/urpmi`, the log files in `/var/log`.

You can put certain packages on 'hold' to prevent them being updated by adding the package names to the file `/etc/urpmi/skip.list`, one name per line. For example, you might want to prevent installing newer `kernel` or `glibc` packages by adding those names to the file.

The file `/etc/urpmi/inst.list` lists all packages which should be installed rather than upgraded. It is preconfigured to exclude all kernel RPMs, to prevent overwriting a running kernel and thus possibly crippling your system. You probably will not need to change this.

When you've installed your system from CDs, these will most likely already be indexed by`urpm`. To add more installation media (other CDs, local directories, remote directories), use`urpmi.addmedia`. Let's say you have a local 'rpms' directory in your

home directory where you keep downloaded RPMs. To add this directory to the urpm database, run (as ***SuperUser***):

```
[root]# urpmi.addmedia local file://home/user_name/rpms
```

'local' is the nick name under which this resource can be referred to in the future. Update the data base with the data from the new 'local' resource:

```
[root]# urpmi.update local
```

Now the packages in 'rpms' are part of the urpm database and taken into account during all urpm operations. Don't forget to run `urpmi.update` if the content of that directory changes (the same goes for all resources with non-static content like FTP directories).

For remote package resources you *have* to provide the relative path to the `hdlist.cz` catalog file on that server (relative from the directory where the RPMs are, that is) in the 'base' directory. You can not add remote resources to urpm unless they provide that dependency file.

```
[root]# urpmi.addmedia contrib
ftp://ftp.sunet.se/pub/Linux/distributions/mandrake/9.1/contrib/RPMS with
../../i586/Mandrake/base/hdlist2.cz
```

There are some tools that can make this easier. If you are running a graphical interface, you can use Mandriva's *Software Manager* to add and remove media, install packages, etc. There is also a web tool called **Easy urpmi** that will help you locate mirrors and generate the `urpmi.addmedia` commands for you to cut and paste.

To remove a resource from urpm, run (as ***SuperUser***):

```
[root]# urpmi.removemedia ${resource_name}
```

`urpm` consists of a set of tools, each performing a certain task (in contrast to ‘rpm’, which works entirely via options).

Installing And Removing Packages

These commands require ***SuperUser*** privileges.

To install a package, use the `urpmi` command. For example, to install the ‘mc’ package:

[root]# urpmi mc

`urpmi` will check if this package is available and either download it (if it’s on a remote resource) or prompt you for the removable medium the package is on or simply install it from the hard disk. If the package needs other packages, `urpmi` will ask you if it is OK to install these, too. If you don’t want to be asked, add the `--auto` option.

In case the package name you provide is ambiguous, `urpmi` will print a list of all matching package names and exit. You can modify this behavior by using the `-a` option:

[root]# urpmi -a gtk

for instance will install *all* packages whose names contain the string ‘gtk’. (Use with caution, this may install more than you expect!)

Another useful option is `-p` which allows to filter packages by what they provide. Example: Let’s say you know you need the ‘libe2p.so.2’ program library, but you do not know which package provides that library:

[root]# urpmi -p libe2p.so.2

makes `urpmi` check which package provides that library and install that package, in this case ‘libext2fs2’.

Of course, you can also use it to simply install a local package file like you would do with`rpm -U`. Notice that `urpmi` always **upgrades** if it finds an newer version of the package than the one currently installed. Sometimes this is not the behavior you want, e.g. when you need two different versions of the same program library. In these cases, you must use `rpm -i`.

To uninstall packages, you use `urpme`:

```
[root]# urpme ${package}
```

If uninstalling the package would break dependencies of other packages, `urpme` asks if these should be removed, too. If you don’t want to be asked, add the `--auto` option. (You should only use this option if you *really* know what you are doing.)

`urpme` also accepts the `-a` option:

[root]# urpme -a gtk

removes all installed packages whose names contain the string ‘gtk’.

Querying Packages

Another area where the `urpm` system really shines is querying, since the `urpm` database also contains information about packages which are not installed. The `urpm` query tool is `urpmf`.***SuperUser*** privileges are not required.

[root]# urpmf ${file}

lists all packages in its database which contain the file ${file}.

`urpmf` supports a lot of options which allow to query certain fields of package information. You want to know what packages containing games are available?

[root]# urpmf –group Games

How big is the ‘pingus’ package?

```
[root]# urpmf --size pingus

 pingus:size:11026299
```

What is this package about?

[root]# urpmf –summary pingus

pingus:summary:Pingus - A free Lemmings clone

Have a look at `man urpmf` for more query options.

There’s another urpm query command called `urpmq`, which is only of limited interest, though.

List packages with names containing ${string} [root]# urpmq ${string}

Show the resource where ${package} is located [root]# urpmq –sources ${package}

List other packages that ${package} depends on [root]# urpmq -d ${package}

List other packages that depend on ${package} [root]# urpmq -r ${package}

Other options are listed in `man urpmq`.

Getting Updates

The urpm system allows you to update your system with the latest security and bug fixes via the command line or even automatically, provided you have added at least one mirror to your `urpmi.cfg` with `urpmi.addmedia`.

The updating command is very simple

[root]# urpmi –auto-select –update

checks all configured resources for updated packages, lists them and asks if they should be installed. If you don’t want to be asked, add the `--auto` option. This option is also convenient when you want you want to run the update via a cron job. Do not forget to run `urpmi.update`to refresh the resource database.

Source : http://www.control-escape.com/linux/lx-swinstall-urpm.html