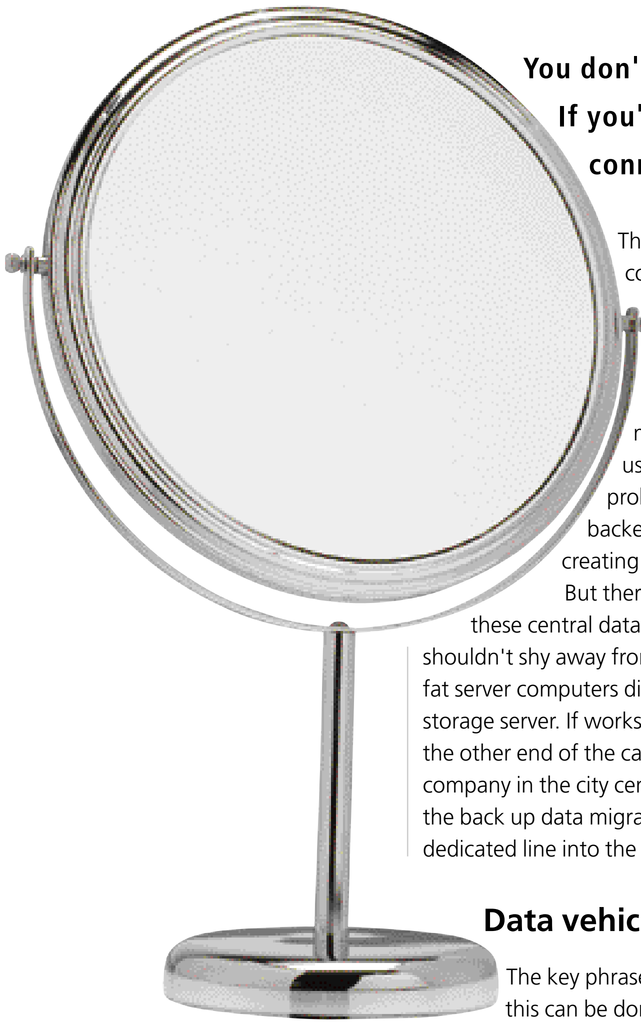
Remote Backups with Rsync and Rdist

SECURITY MIRRORING

PATRICIA JUNG



You don't have to use classic data back up methods. If you're part of a local network or have an Internet connection, you could use remote back up.

They're the stars of any large computing centre – the robots which equip the tape drives of commercial data storage facilities. While there's no shortage of memory space there, smaller users are still faced with probing the mass of data to be backed up and actually creating copies.

But there is another principle of these central data back up systems which you shouldn't shy away from. There may be a couple of fat server computers directly attached to the storage server. If workstations in the next room, at the other end of the campus or from the client company in the city centre have to be backed up, the back up data migrate via LAN, dial-up or dedicated line into the storage system.

Data vehicles

The key phrase here is remote backup, and this can be done even when one has a small LAN or sufficient space on another computer on the Internet. The only requirement is that there is server software running on the backup computer which allows log ons from remote computers.

Nowadays this will be a secure shell server (*sshd*). On some machines, you will also still find *Remote-Sh*ell-*Daemons rshd*, but since this means the data is transferred unencrypted, this should only be considered in a well-secured LAN, in which all users are trustworthy.

To *mirror* the data (where synchronous copies of original data are made on separate media) there are two possible programs: *rdist* and *rsync*. The original *rdist* is no longer being actively developed, but there are still projects such as *freerdist* (*http://freshmeat.net/projects/freerdist/*), which can at least show last-amended data. And not very much will change in terms of functionality in future – which can certainly be an advantage.

In the case of *rsync* (*http://rsync.samba.org/rsync/*) the authors are more energetic. This program has the advantage that, where files have changed, it only transfers the changes. *rdist* sends the complete file again. If you want to keep the transfer volume as low as possible, this is a crucial criterion for decision-making.

Otherwise it really depends on your individual taste. Neither has yet been the serious target of GUI programmers, so you will still have to get to grips with the syntax, which takes some getting used to.

Since we devoted a long article to *rsync* in Answer Girl (Linux Magazine issue 9, p. 84 ff), we will now cover the alternative product.

Remote distribution

Before you can configure the data mirroring, it is worthwhile making sure that *rdist* is installed not only on the system with the data to be backed up, but also on the destination system. The *rdist* client may not necessarily be installed, but the *rdist* daemon *rdistd* must, because this is called on when it comes to data comparison.

If *rdistd* is not on the search path to the destination account, it is best to note down where it is.

On the computer with the data to be duplicated, you must take the *rdist* client by the hand. In the first instance, this will only know the *P*ath to the secure shell client *ssh* and perhaps the name and path of its configuration file:

```
/tmp$ rdist -P 'which ssh' -f ~/distfile
```

If the *rdistd* cannot be found on the destination system without precise path specification, there is still the option *-p /path/to/backupcomputers/rdistd*.

If there is a file named *distfile* or *Distfile* in the current directory, the *-f* option can also be left out:

```
~$ rdist -P /usr/bin/ssh
```

But this file is not to be sniffed at, since it contains all the details on what, how and to where it is to be duplicated. Several entries are possible here, so that different copy rules can be specified for various directories. If you like, you can back up the data for your thesis on a uni account, while love letters would be better backed up on the second computer in the home LAN.

Each entry begins with a name for the following rule and a colon. The former can be anything you like, but can only be *one* word without a space. Then comes the specification of *what* is to be copied. This can be directories or just individual files. If there are several specifications separated by colons, everything must be enclosed in a pair of round brackets.

Next comes a stylised arrow, ->, pointing to the address of the destination computer. If you have a different username on the other system, this is written in front, as with an email address. In that case, a @ separates user name and host name.

Thus, a *distfile* with the following content ...

```
private: ( ~/.netscape/bookmarks.html \
        ~/private \
        ~/letters ) -> trish@192.168.1.249

thesis: /home/trish/thesis -> lillegroenn.tr➘
ish.de
```

... is saying that the directories *~/private* and *~/letters* together with the Netscape bookmarks should land in the account of *trish* on the computer with the IP address *192.168.1.249*, while the directory */home/trish/thesis* with all its sub-directories, should be shovelled onto the computer *lillegroenn.trish.de*.

Destination anywhere?

What's missing now is the Where To on the destination computers. This is specified with *install*:

```
      install /mnt/backup;
```

ensures that the respective data lands under */mnt/backup* on the destination system. If there isn't one, *rdist* (to be precise, the *rdistd* called up by it on the destination computer) also makes this directory. What matters here is the semicolon at the command end.

There are a few commands such as *install*, which modify the action of *rdist*; we shall pick out two more at this point, which are of general interest with respect to backups: *except_pat* excludes files and directories from the backup which corresponds to the pattern specified.

```
      except_pat tmp;
```

ensures that *tmp* directories, but also files such as *wtmp* or *chapter1.tmp* are not backed up at the same time. One can specify several patterns in round brackets, and regular expressions can be used to a certain degree. So

```
      except_pat ( \\.tgz\$ [Tt][mM][pP] );
```

ensures that all files ending in *.tgz* (\$ stands for the end of the file name) and all files/directories with *tmp* in any combination of upper and lower case letters are excluded from the backup. Since the dot is meant, not as a regular expression for any symbol, but as a dot, there must be a \ in front, and as this, too, is a special symbol, another backslash is placed in front of that.

The command

```
      notify pjung@linux-magazine.co.uk;
```

in turn ensures that *pjung@linux-magazine.co.uk* receives an e-mail, in which *rdist* reports on the work performed.

Equipped with all these options, *distfile* then looks e.g. as in Listing 1. Comments are – as usual in the shell – preceded by a #.

Doppelgängers

Provided you leave the back up space in peace, everything is hunky-dory. Nevertheless, now and then you change one file or another and do not want the back up to ruthlessly overwrite these changes. To inspire in *rdist* a little consideration at this point for files which are newer on the destination system than on the source computer, the quick and easy option *-o younger* is added to the *rdist* call up.

And a contrary approach is also possible: If you want to ruthlessly destroy everything on the back up system which does not exist on the original system, specify the option *-o remove* on the command line.

Of course, that does not bring us to the furthest limit of fine tuning. The *rdist* man page can become your constant companion when planning a backup. ■

Listing 1: Example of an *rdist* distfile

```
private: ( ~/private ~/letters ) -> trish@192.168.1.249
      install /mnt/backup/private;
      except_pat tmp;

thesis: /home/trish/thesis -> lillegroenn.trish.de
      install ~/backup;
      except_pat ( \\.tgz\$ [Tt][mM][pP] );
      notify pjung@linux-magazine.co.uk;

# Copy the Netscape bookmarks on the spot
bookmarks: ~/.netscape/bookmarks.html -> lillegroenn.trish.de
        install ~/.netscape/bookmarks.html;
```