What's so great about rsync? First, it's designed to speed up file transfer by copying the differences between two files rather than copying an entire file every time. For example, when I'm writing this article, I can make a copy via rsync now and then another copy later. The second (and third, fourth, fifth, etc.) time I copy the file, rsync copies the differences *only*. That takes far less time, which is especially important when you're doing something like copying a whole directory offsite for daily backup. The first time may take a long time, but the next will only take a few minutes (assuming you don't change that much in the directory on a daily basis).

Another benefit is that rsync can preserve permissions and ownership information, copy symbolic links, and generally is designed to intelligently handle your files.

You shouldn't need to do anything to get rsync installed -- it should be available on almost any Linux distribution by default. If it's not, you should be able to install it from your distribution's package repositories. You will need rsync on both machines if you're copying data to a remote system, of course.

When you're using it to copy files to another host, the rsync utility typically works over a remote shell, such as Secure Shell (SSH) or Remote Shell (RSH). We'll work with SSH in the following examples, because RSH is not secure and you probably don't want to be copying your data using it. It's also possible to connect to a remote host using an rsync daemon, but since SSH is practically ubiquitous these days, there's no need to bother.

Getting to know rsync

The basic syntax for rsync is simple enough -- just run rsync [options] *source destination* to copy the file or files provided as the
source
argument to the
destination
.

So, for example, if you want to copy some files under your home directory to a USB storage device, you might use rsync -a /home/user/dir/ /media/disk/dir/. By the way, "/home/user/dir/" and "/home/usr/dir" are *not* the same thing to rsync. Without the final slash, rsync will copy the directory in its entirety. With the trailing slash, it will copy the contents of the directory but won't

recreate the directory. If you're trying to replicate a directory structure with rsync, you should omit the trailing slash -- for instance, if you're mirroring /var/www on another machine or something like that.

In this example, I included the archive option (-a), which actually combines several rsync options. It combines the recursive and copy symlinks options, preserves group and owner, and generally makes rsync suitable for making archive copies. Note that it *doesn't* preserve hardlinks; if you want to preserve them, you will need to add the hardlinks option (-H).

Another option you'll probably want to use most of the time is verbose (-v), which tells rsync to report lots of information about what it's doing. You can double and triple up on this option -- so using -v will give you some information, using -vv will give more, and using -vvv will tell you everything that rsync is doing.

rsync will move hidden files (files whose names begin with a .) without any special options. If you want to exclude hidden files, you can use the option --exclude=".*/". You can also use the --exclude option to prevent copying things like Vim's swap files (.swp) and automatic backups (.bak) created by some programs.

Making local copies

Suppose you have an external USB or FireWire drive, and you want to copy data from your home directory to your external drive. A good way to do this would be to keep all your important data under a single top-level directory and then copy it to a backup directory on the external drive using a command like:

```
rsync -avh /home/usr/dir/ /media/disk/backup/
```

If you want to make sure that local files you've deleted since the last time you ran rsync are deleted from the external system as well, you'll want to add the --deleted option, like so:

```
rsync -avh --delete /home/user/dir/ /media/disk/backup
```

Be very careful with the delete option; with it, you can whack a bunch of files without meaning to. In fact, while you're getting used to rsync, it's probably a good idea to use the --dry-run option with your commands to run through the transfer first, without actually copying or synching files. If you do start an rsync transfer and realize that you've botched the command in some way that might result in the destruction of data, press Ctrl-c immediately to terminate the transfer. Some files may be gone, but you may be able to save the rest.

Making remote copies

What if you want to copy files offsite to a remote host? No problem -- all you need to do is add the host and user information. So, for instance, if you want to copy the same directory to a remote host, you'd use:

rsync -avhe ssh --delete */home/user/dir/ user@remote.host.com:dir/*

If you want to know how fast the transfer is going, and how much remains to be copied, add the --progress option:

rsync --progress -avhe ssh --delete

/home/user/dir/ user@remote.host.com:dir/

If you don't want to be prompted for a password each time rsync makes a connection -- and you don't -- make sure that you have rsync set up to log in using an SSH key rather than a password. To do this, create an SSH key on the local machine using ssh-keygen -t dsa, and press Enter when prompted for a passphrase. After the key is created, use ssh-copy-id -i .ssh/id_dsa.pub *user@remote.host.com* to copy the public key to the remote host.

What if you need to bring back some of the files you copied using rsync? Use the following syntax:

rsync -avze ssh remote.host.com:/home/user/dir/ /local/path/

The z option compresses data during the transfer. If the file you are copying exists on the local host, then rsync will just leave it alone -- the same as if you were copying files to a remote host.

Wrapping it up with a script

Once you've figured out what directory or directories you want to sync up, and you've gotten the commands you need to sync everything, it's easy to wrap it all up with a simple script. Here's a short sample:

rsync --progress -avze ssh --delete /home/user/bin/ user@remote.host.com:bin/rsync --progress -avze ssh --delete /home/user/local/data/ user@remote.host.com:local/data/rsync --progress -avze ssh --delete /home/user/.tomboy/ user@remote.host.com:/.tomboy/

Use the --progress option if you're going to be running rsync interactively. If not, there's no need for it.

If you look at the rsync man page , you can easily be confused. However, after a little practice with rsync, you'll find that it's not hard to set up rsync jobs that will help you prepare for the day that your disk drive craps out and you need access to your data right away.