How to move files between systems using scp and rsync

There are many tools for moving files between two computers via the Linux command line, but the scp and rsync commands are two of the most popular choices, covering almost every use case.

There are many tools for moving files between two computers using the Linux command line, but the scp and rsync commands are two of the most popular options, covering almost every use case. This article will show you how and when to use the scp and rsync commands in a Linux environment.

What is the difference between scp and rsync?

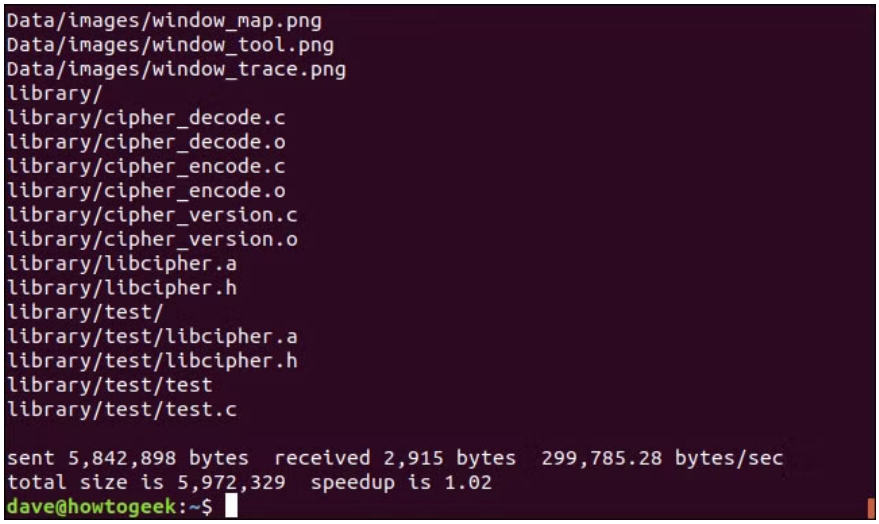
Both scp and rsync can support moving files between two computers, and both can do so very securely. The main difference between the two commands is not their purpose, but how they do the job.

The scp command stands for " **secure copy** " and in many ways it works just like the standard unix cp or " **copy** " command, but between two computers. " **secure** " in this case refers to SSH, the protocol the command uses to communicate with the remote computer.

This command is as simple as its name suggests, reading files from one computer and writing them to another. This simplicity fits the UNIX philosophy of "doing one thing well".

The rsync command does a similar job, but takes a more subtle approach. The difference is in the name. rsync will copy a file to the other computer if it doesn't already exist, any future operations will just compare the differences between the files on each computer.

If you're only moving a small number of files, this difference won't be very noticeable. However, if you're using rsync as part of a backup script, running it on files that have changed can save a lot of time and bandwidth. Of course, this requires rsync to be installed on both the local and remote computers, whereas the scp command only needs to be installed on the computer running the command.



This isn't the only difference. The rsync command is also more 'durable' and can re-establish a connection if it drops mid-transfer. Combined with the ability to only transfer the data that differs, this makes it much more suitable for tasks that require transferring large amounts of data.

rsync also has similarities with scp. For example, it can also connect over SSH and you can use SSH keys to connect to the remote machine with any command.

How to transfer files using scp

Using the scp tool to move single files between computers is very simple. The command can be as basic as:

```
scp username@remotehost:/path/to/directory
```

To copy the file ' **foo.txt** ' to the remote computer, you can use the command below:

```
scp foo.txt kris@some-computer:/home/kris/
```

To use the scp command to copy that file back to your computer (for whatever reason), you can use the following command:

```
scp kris@some-computer:/home/kris/foo.txt
```

Of course, in many cases you'll want to move entire directories of files between computers. Fortunately, this is as simple with scp as it is with the standard cp command. Just pass the -r flag to copy a directory recursively:

```
scp -r dotfiles kris@some-computer:/home/kris/
```

Or to copy a directory recursively from a remote computer, use the following command:

```
scp -r kris@some-computer:/home/kris/dotfiles
```

How to transfer files using rsync

rsync has a lot of built-in functionality that makes handling basic tasks fairly simple. The basic command to copy files to a remote computer is as follows:

```
rsync  local-filename > user@remotehost: remote-filename >
```

To copy the same file 'foo.txt' from the scp example above, you would use the following command:

```
rsync foo.txt kris@some-computer:foo.txt
```

In the background, rsync will attempt to use an SSH connection and will ask you for a username and password. Once you are authenticated, the transfer will begin.

Conversely, copying the file back to your local computer is just as simple:

```
rsync kris@some-computer:/home/kris/foo.txt .
```

Of course, doing it this way doesn't really take advantage of the full power of rsync and its various options. rsync has many flags to set different options, but one flag you'll want to use is -a or "--archive". Combining a few options, rsync can recursively scan directories, preserve ownership and permissions, and copy symbolic links as symbolic links.

Given the common use cases for rsync, it makes sense to put these options in an easily accessible flag. For most users' rsync needs, the -a flag is the usual case. For example, to synchronize a directory with a remote computer:

```
rsync -av ~/dotfiles kris@some-computer:/home/kris/dotfiles
```

The second flag in the above command is ' **-v** ' **-verbose** '. This flag will give you more detailed information about the progress of the command as it runs.

Depending on your internet connection, you can speed up your transfer by compressing the data during transfer. This keeps the data uncompressed on both your local and remote machines, but compresses the data during transfer. To do this, use the **-z** flag :

```
rsync -azv ~/dotfiles kris@some-computer:/home/kris/dotfiles
```

Again, this isn't guaranteed to speed up data transfers, but it's still worth considering, especially for frequent tasks like backup scripts or other jobs that take a long time.

When to use scp or rsync?

Both the scp and rsync commands have their own strengths and weaknesses, but in everyday use, determining which command is best often depends on how you use them. There is a fairly simple rule of thumb that you can use to see which command to use in a given situation.

If you're using a manual command to move files, it's generally better to use scp. It's simpler and easier to understand, and you're probably already at least familiar with the basic flags used with the cp command.

On the other hand, if you're writing a script, rsync might be the best choice. It's better at handling unexpected network failures, and the deeper nature of the sync means it has more powerful options than the relatively simple scp utility.

Both commands are worth knowing and are useful to have in your essential Linux command toolkit.