To exclude a list of files specified as command-line arguments use the `-e` option. For example:

```
$ sudo mksquashfs /etc test.squashfs -e /etc/passwd /etc/shadow
```

The `-e` option is used to exclude `passwd` and `shadow` files.

It is also possible to specify a list of exclude files given in a file with `-ef` as follows:

```
$ cat excludelist
/etc/passwd
/etc/shadow

$ sudo mksquashfs /etc test.squashfs -ef excludelist
```

If we want to support wildcards in excludes lists, use `-wildcard` as an argument.

Backup snapshots with rsync

Backing up data is something that most sysadmins need to do regularly. In addition to backing up local files, we may need to backup data from a web server or from remote locations. `rsync` is a command that can be used to synchronize files and directories from one location to another while minimizing data transfer using file difference calculations and compression. The advantage of `rsync` over the `cp` command is that `rsync` uses strong difference algorithms. Additionally, it supports data transfer across remote machines. While making copies, it compares the files in the original and destination locations and will only copy the files that are newer. It also supports compression, encryption, and a lot more. Let us see how to work with `rsync`.

How to do it...

Let's see how to copy files and create backups with `rsync`:

1. To copy a source directory to a destination use:

   ```
   $ rsync -av source_path destination_path
   For example,
   $ rsync -av /home/slynux/data slynux@192.168.0.6:/home/backups/
   data
   ```

   In this command:

   - `-a` stands for archiving
   - `-v` (verbose) prints the details or progress on `stdout`

   The above command will recursively copy all the files from the source path to the destination path. We can specify paths as remote or local paths.

2. In order to backup data to a remote server or host, use:

```
$ rsync -av source_dir username@host:PATH
```

   To keep a mirror at the destination, run the same `rsync` command scheduled at regular intervals. It will copy only changed files to the destination.

3. Restore the data from the remote host to `localhost` as follows:

```
$ rsync -av username@host:PATH destination
```

The `rsync` command uses SSH to connect to the remote machine, hence you should provide the remote machine's address in the format `user@host`, where user is the username and host is the IP address or host name attached to the remote machine. `PATH` is the path on the remote machine from where the data needs to be copied.

Make sure that OpenSSH server is installed and running on the remote machine. Additionally, to prevent the prompt for a password for the remote machine, see the recipe "*Password-less auto-login with SSH*" from Chapter 7.

4. Compressing data while transferring through the network can significantly optimize the speed of the transfer. We can use the `rsync` option `-z` to specify to compress data while transferring through a network. For example:

```
$ rsync -avz source destination
```

5. Synchronize one directory to another directory as follows:

```
$ rsync -av /home/test/ /home/backups
```

   This command copies the source (`/home/test`) to an existing folder called backups.

6. Copy a full directory inside another directory as follows:

```
$ rsync -av /home/test /home/backups
```

This command copies the source (`/home/test`) to a directory named backups by creating that directory.

For the PATH format, if we use `/` at the end of the source, `rsync` will copy contents of that end directory specified in the `source_path` to the destination.

If `/` is not present at the end of the source, `rsync` will copy that end directory itself to the destination.