The **user** is the owner of the file. The **group** is the collection of users (as defined by the system administrator) that are permitted some access to the file. **Others** are any entities other than the user or group owner of the file.

Permissions of a file can be listed by using the `ls -l` command:

```
-rw-r--r-- 1 slynux slynux  2497  2010-02-28 11:22 bot.py
drwxr-xr-x 2 slynux slynux  4096  2010-05-27 14:31 a.py
-rw-r--r-- 1 slynux slynux  539   2010-02-10 09:11 cl.pl
```

The first column of the output specifies the following, with the first letter corresponding to:

- `-` – if it is a regular file
- `d` – if it is a directory
- `c` – for a character device
- `b` – for a block device
- `l` – if it is a symbolic link
- `s` – for a socket
- `p` – for a pipe

The rest can be divided into three groups of three letters each (--- --- ---). The first --- three characters correspond to the permissions of the user (owner), the second set of three characters correspond to the permissions of the group, and the third set of three characters correspond to the permissions of others. Each character in the nine-character sequence (nine permissions) specifies whether permission is set or unset. If the permission is set, a character appears in the corresponding position, otherwise a `-` character appears in that position, which means that the corresponding permission is unset (unavailable).

Let's take a look at what each of these three character sets mean for the user, group, and others:

- **User** (permission string: `rwx------`): The first letter in the three letters specifies whether the user has read permission for the file. If the read permission is set for the user, the character `r` will appear as the first character. Similarly, the second character specifies write (modify) permission (`w`) and the third character specifies whether the user has execute (`x`) permission (the permission to run the file). The execute permission is usually set for executable files. The user has one more special permission called **setuid** (`S`), which appears in the position of execute (`x`). The setuid permission enables an executable file to be executed effectively as its owner, even when the executable is run by another user.

  An example for a file with setuid permission set is `-rwS------`.

The read, write, and execute permissions are also applied to the directories. However, the meanings of read, write, and execute permissions are slightly different in the context of directories as follows:

- The read permission (`r`) for the directories enables reading the list of files and subdirectories in the directory
- The write permission (`w`) for a directory enables creating or removing files and directories from a directory
- The execute permission (`x`) specifies whether the access to the files and directories in a directory is possible or not

- **Group** (permission string: `---rwx---`): The second set of three characters specifies the group permissions. The interpretation of permissions `rwx` is the same as the permissions for the user. Instead of setuid, the group has a **setgid** (`S`) bit. This enables the item to run an executable file with an effective group as the owner group. But the group, which initiates the command, may be different.

  An example of group permission is `----rwS---`.

- **Others** (permission string: `------rwx`): Other permissions appear as the last three character set in the permission string. Others have the same read, write, and execute permissions as the user and group. But it does not have permission `S` (such as setuid or setgid).

Directories have a special permission called a **sticky bit**. When a sticky bit is set for a directory, only the user who created the directory can delete the files in the directory, even if the group and others have write permissions. The sticky bit appears in the position of execute character (`x`) in the others permission set. It is represented as character `t` or `T`. The `t` character appears in the position of `x` if the execute permission is unset and the sticky bit is set. If the sticky bit and the execute permission are set, the character `T` appears in the position of `x`. For example:

```
------rwt , ------rwT
```

A typical example of a directory with sticky bit turned on is `/tmp`.

In each of the `ls -l` output lines, the string `slynux slynux` corresponds to the owned user and owned group. Here, the first `slynux` is the user and the second `slynux` is the group owner.

How to do it...

In order to set permissions for files, we use the `chmod` command.

Assume that we need to set permission: `rwx rw- r—`.