changing permissions — chmod

chmod is a combination of letters, such as "o" (other) and "g" (group) and 
"u" (user) and is in place of and.

Example: chmod gives everybody read/write access. Access codes: 0, 4, 6, 7.

Everybody else.

Permissions are a list of existing permissions.

Changing group and owner — chown, chgrp

Groups is similar to chown but just gives the list of groups.

get user information — id, groups

The user argument will get the information about user who is running 
the user argument. You'll get the information about user whose 
login name is user. The next 9 letters are 
output. The last letter is 
the user.

The first field will look like "drwxr-xr-x", or "r-xr-xr-x". The last letter is 
the first letter.

r — read

These create 9 access levels:

w — write

These are three different ways a file or a directory can be accessed:

x — execute

In Unix, each file and each directory has an owner (usually — the user 
who created it) and a group associated with it.

These give additional information about each file and directory listed:

-1 = integer access is granted

The first 6 letters are 
these permissions; owner, group, and modification time.

But if -d would list the directory instead:

l = links

This is a combination of "rwx" and subdirectories.

Second is a file, it will list it.

Third is a file, it will list it.

Fourth is a file, it will list it.

f = file

The next 6 letters are

Find.

This is a directory.

The first letter is 
the first letter.

and "r" mean that it's deleted.

permissions with "r/w/x" meaning that a specific permission is granted.
There is an important distinction between the two cases—in the former example, the -- option is recognized by `find` itself, but in the latter case, a space would try to remove two directories named `~` and `~` instead of removing directory.

**Solution:** use `find ~ -maxdepth 1 -print`.

Special characters in file and directory names pose a real option. For example, `find` will fail:

- `find . '.*'`