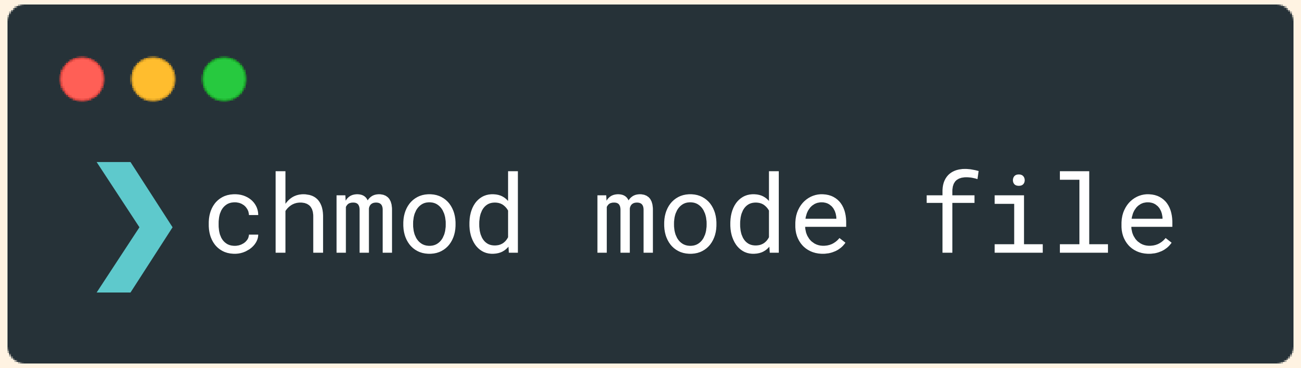
Altering Permissions

chmod

To change the permissions of a file or directory, we can use the chmod command (change mode).

To use chmod to alter permissions, we need to tell it:

- **Who** we are changing permissions for
- **What** change are we making? Adding? Removing?
- **Which** permissions are we setting?



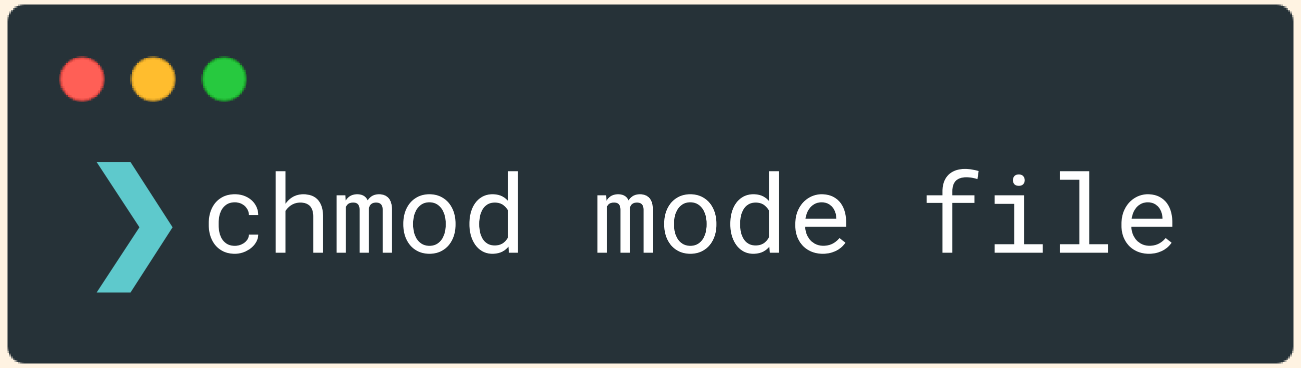
```
❯ chmod mode file
```

chmod

When specifying permissions with chmod, we use a special syntax to write permission statements.

First, we specify the "who" using the following values:

- **u** - user (the owner of the file)
- **g** - group (members of the group the file belongs to
- **o** - others (the "world"
- **a** - all of the above



```
❯ chmod mode file
```

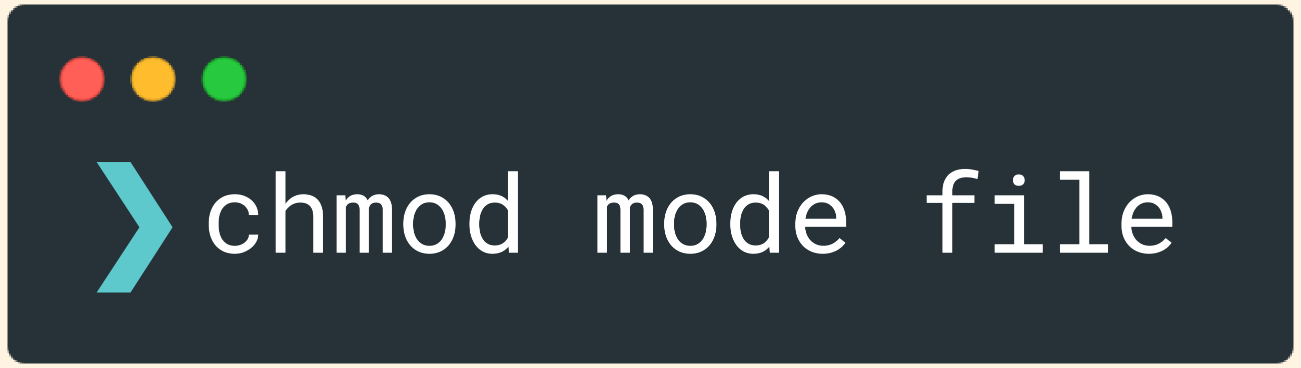
chmod

Next, we tell chmod "what" we are doing using the following characters:

- - (minus sign) removes the permission
- + (plus sign) grants the permission
- = (equals sign) set a permission and removes others

Finally, the "which" values are:

- r - the read permission
- w - the write permission
- x - the execute permission



```
❯ chmod mode file
```

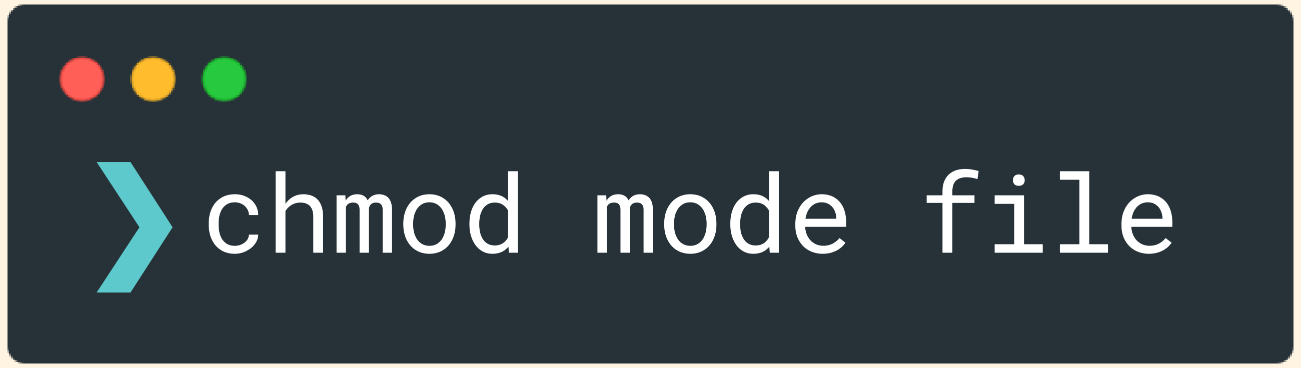
All Togther Now

Next, we tell chmod "what" we are doing using the following characters:

- - (minus sign) removes the permission
- + (plus sign) grants the permission
- = (equals sign) set a permission and removes others

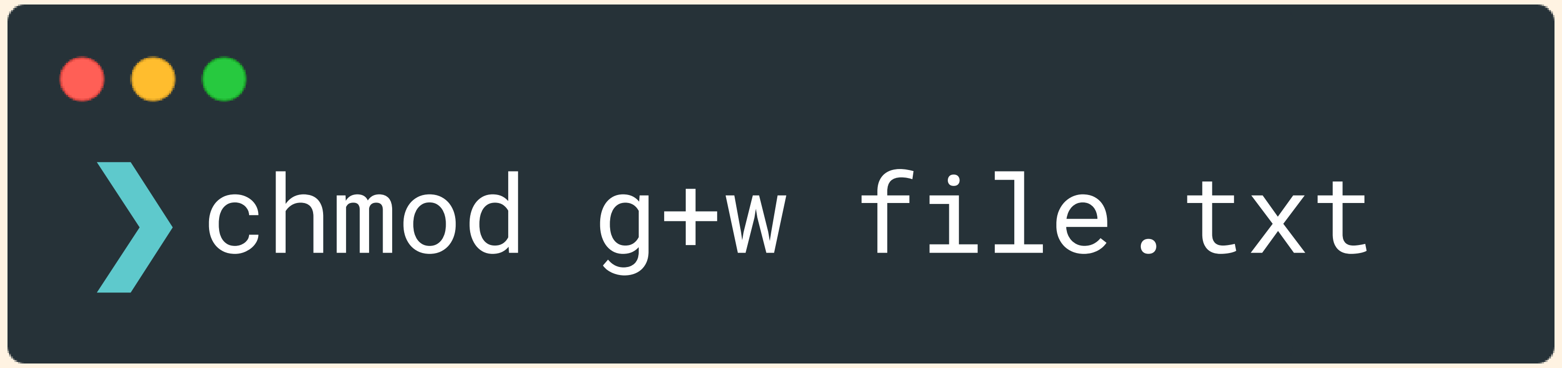
Finally, the "which" values are:

- r - the read permission
- w - the write permission
- x - the execute permission



```
❯ chmod mode file
```

	Owner	Group	World
-	rw-	r--	r--

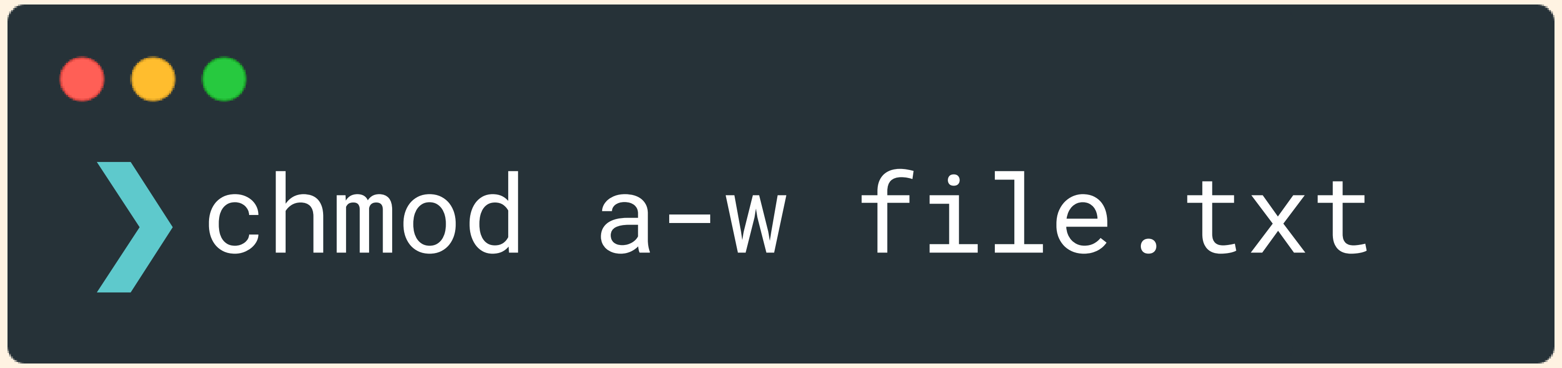


```
❯ chmod g+w file.txt
```

Add write permissions to the group

	Owner	Group	World
-	rw-	rw-	r--

	Owner	Group	World
-	rw-	rw-	r--

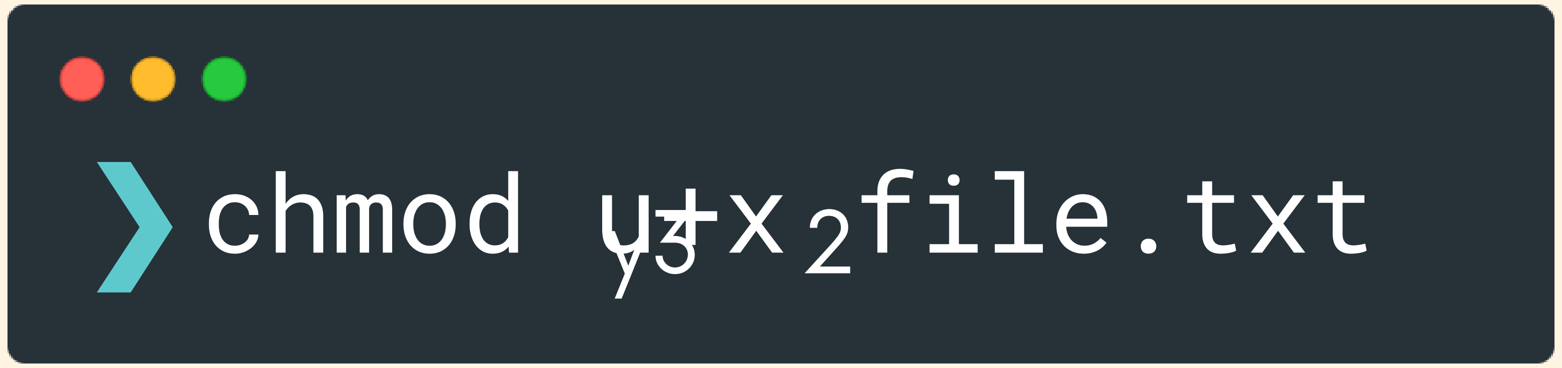


```
❯ chmod a-w file.txt
```

Remove write permissions from all

	Owner	Group	World
-	r--	r--	r--

	Owner	Group	World
-	rw-	rw-	r--

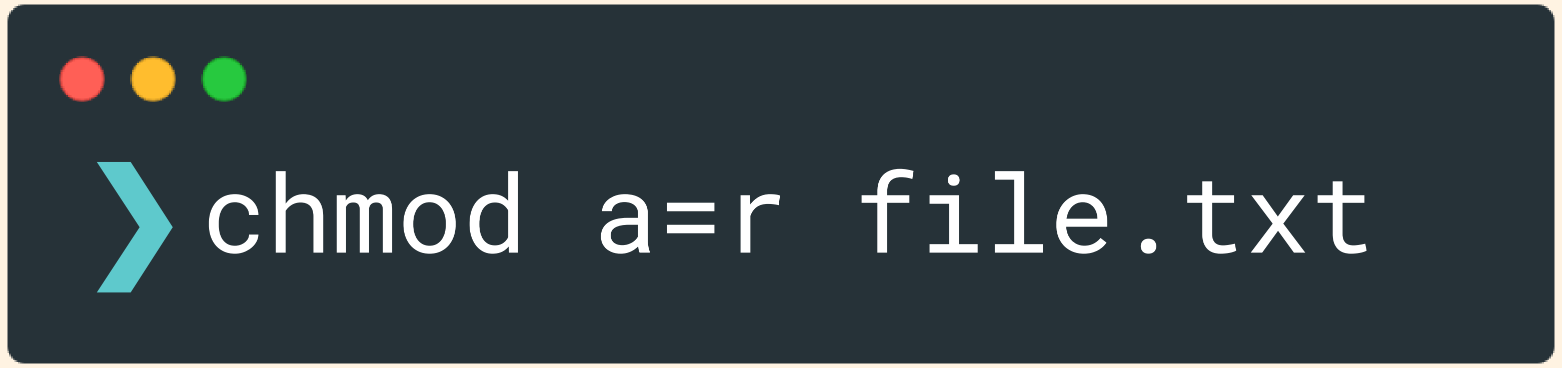


```
❯ chmod u+x file.txt
```

Add executable permissions for owner

	Owner	Group	World
-	rwx	rw-	r--

	Owner	Group	World
-	rwx	rwx	r--



```
❯ chmod a=r file.txt
```

Set permissions to read ONLY for all.

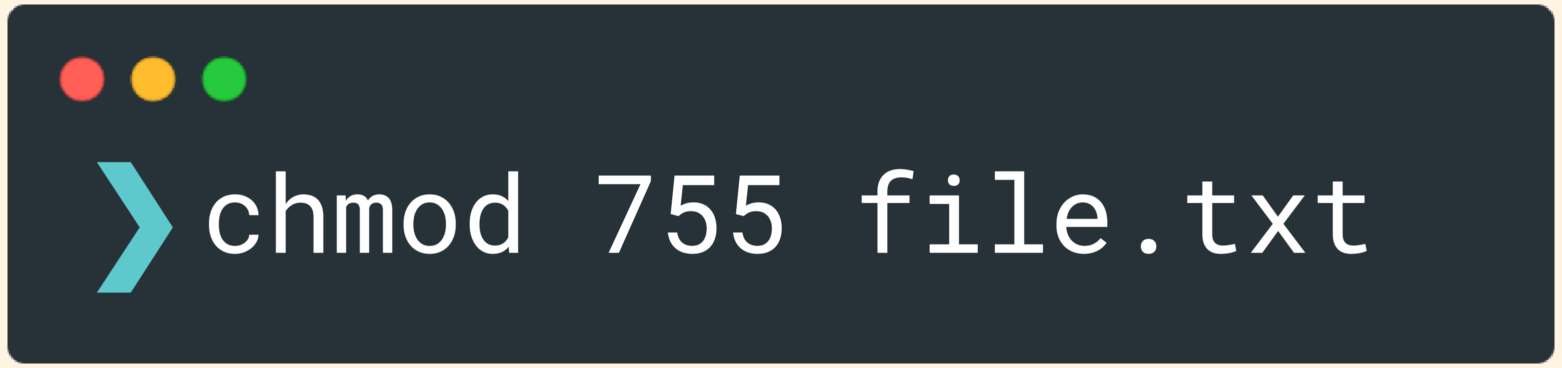
	Owner	Group	World
-	r--	r--	r--

chmod octals

chmod also supports another way of representing permission patterns: octal numbers (base 8). Each digit in an octal number represents 3 binary digits.

Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx



```
❯ chmod 755 file.txt
```

Owner	Group	World
111	101	101

Owner	Group	World
rwx	r-x	r-x

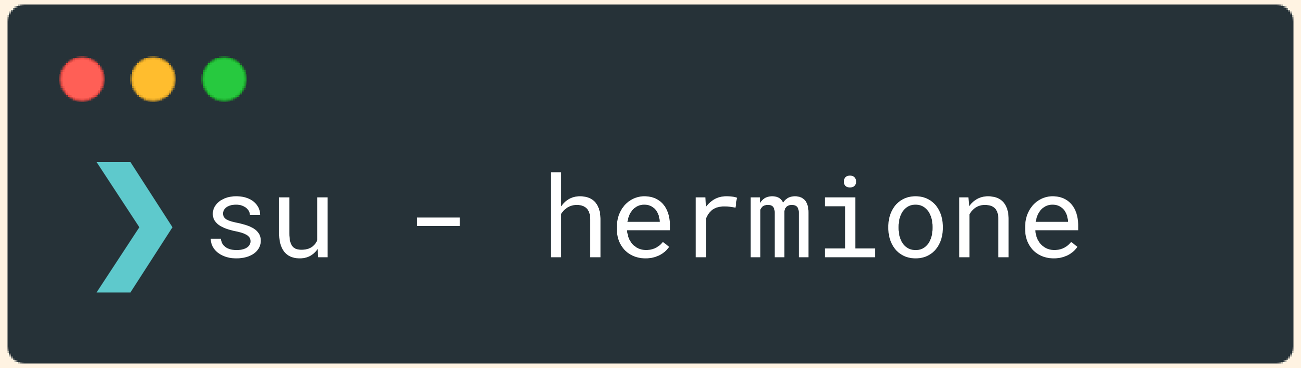
Changing Our Identity

There may be times we want to start a shell as another user, from within our own shell session.

We can use the su command to do just that.

For example, **su - hermione** would create a new login shell for the user hermione. We would need to enter Hermione's passsword.

To leave the session, type **exit**.



```
❯ su - hermione
```

Root User

In Linux systems, there is a super user called root. The root user can run any command and access any file on the machine, regardless of the file's actual owner.

The root user has tons of power and could easily damage or even destroy the system by running the wrong commands!

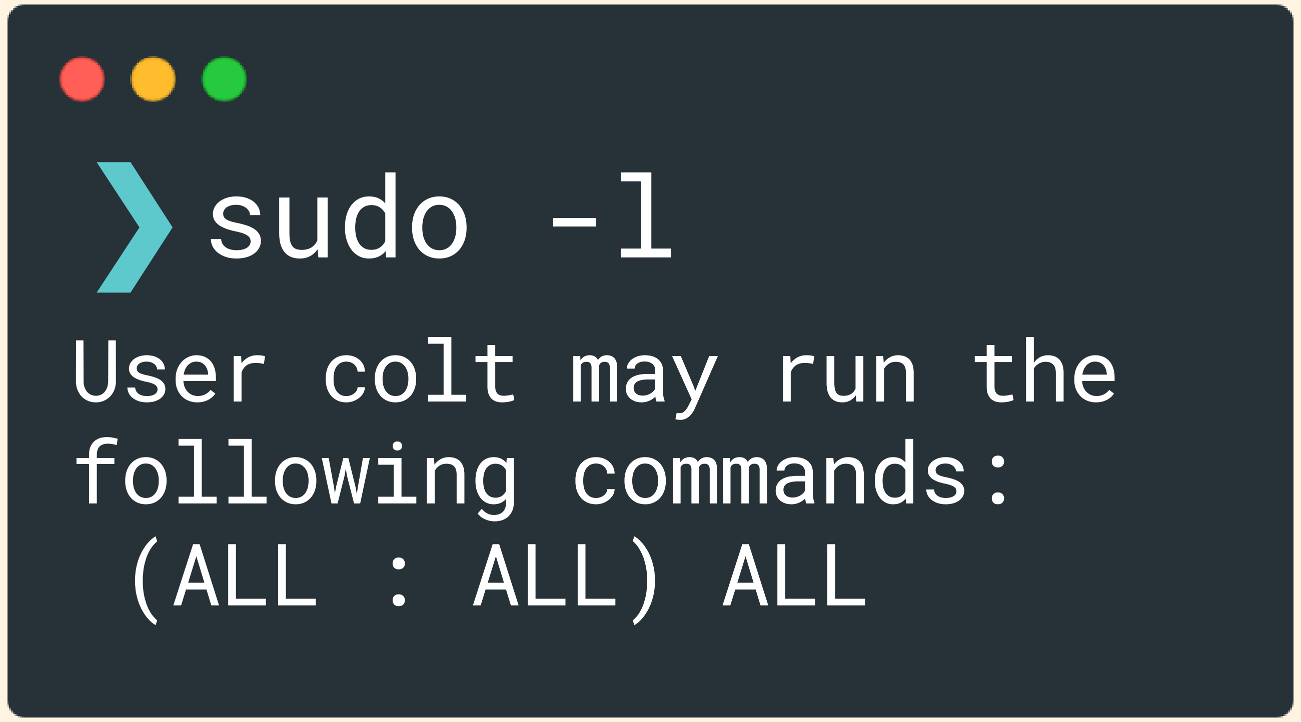
For this reason, Ubuntu locks the root user by default.

Sudo

Even if the root user is locked by default, we can still run specific commands as the root user by using the **sudo** command.

Individual users are granted an "allowed" list of commands they can run as the super user.

Run sudo -l to see the permitted commands for your particular user.



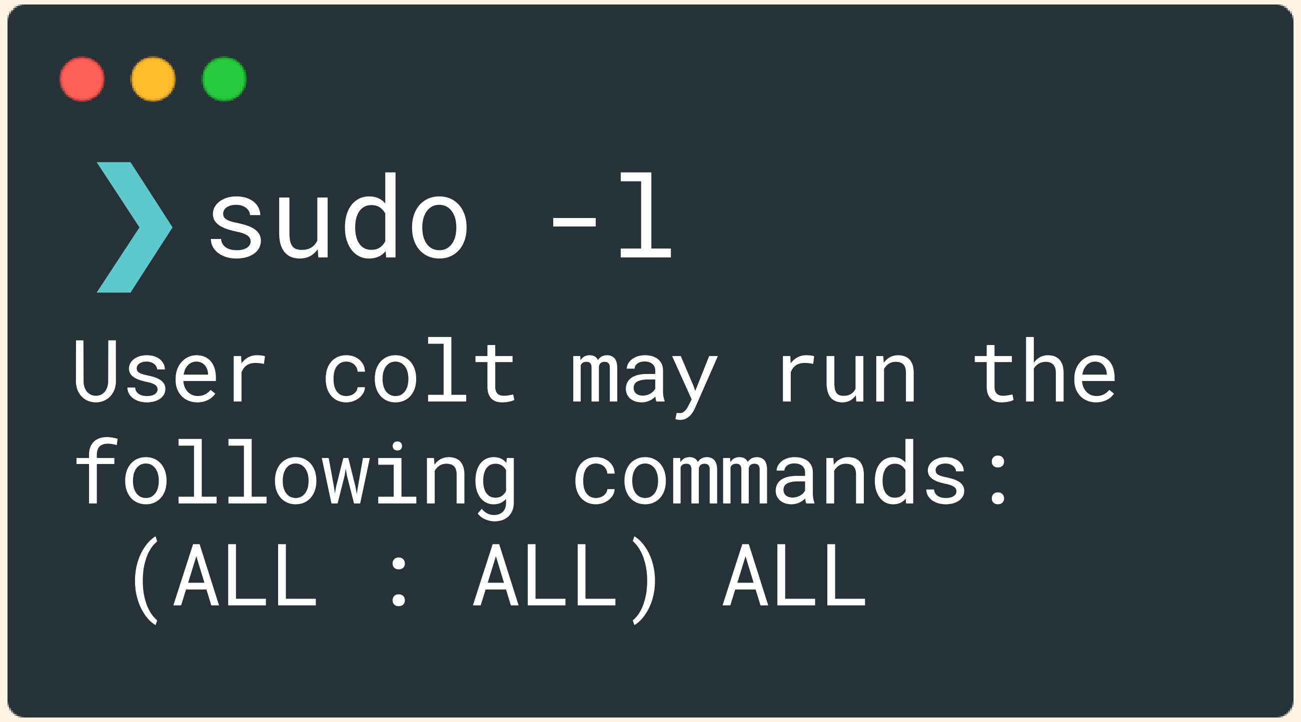
```
❯ sudo -l
User colt may run the
following commands:
 (ALL : ALL) ALL
```

Sudo + Ubuntu

By default, Ubuntu disables logins to the root account. Instead, the initial user is granted full access to all superuser priveleges:

"User colt may run the following commands:
 (ALL : ALL) ALL"

Subsequent users won't have full sudo privileges by default, but the original user can grant them.



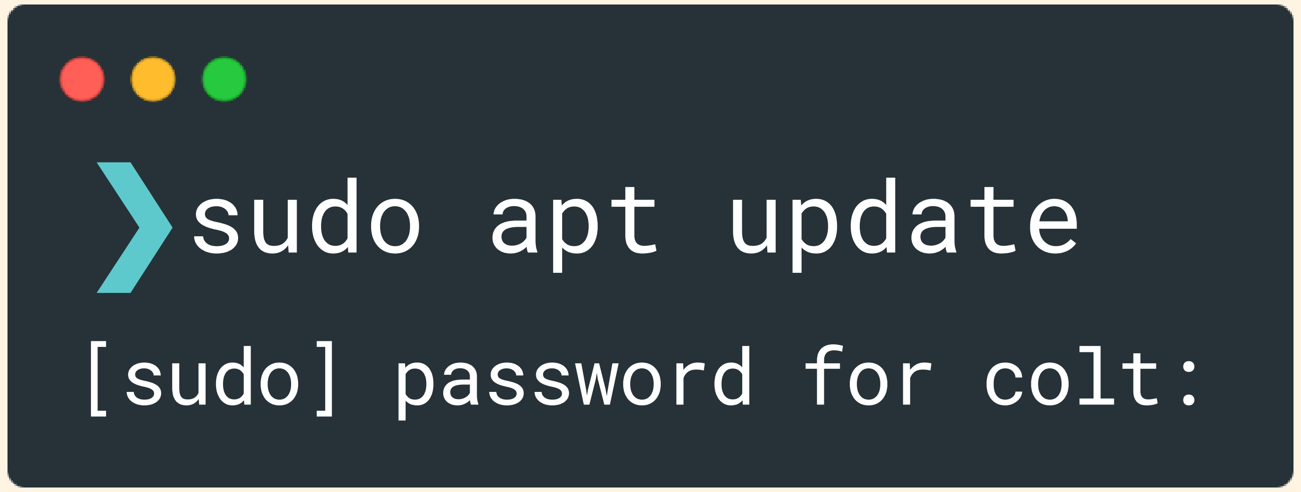
```
❯ sudo -l
User colt may run the
following commands:
 (ALL : ALL) ALL
```

Sudo

To run a command as the root user, prefix it with sudo. You will then need to enter the password for **your account**.

For example to update Ubuntu, I would need to run apt update. However, I can't do this as my "regular" user, as it's something that impacts all users.

Instead, I need to run the command as the root user using sudo apt update

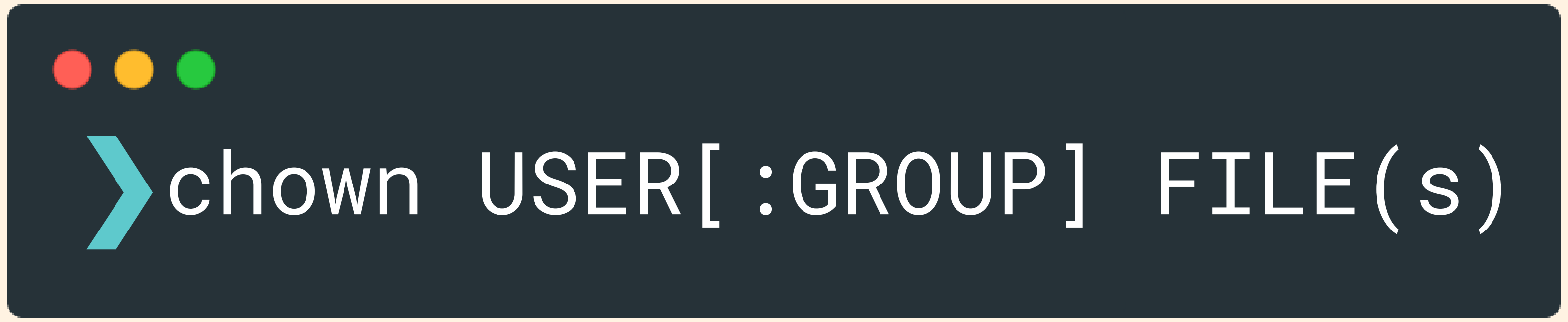


```
❯sudo apt update
[sudo] password for colt:
```

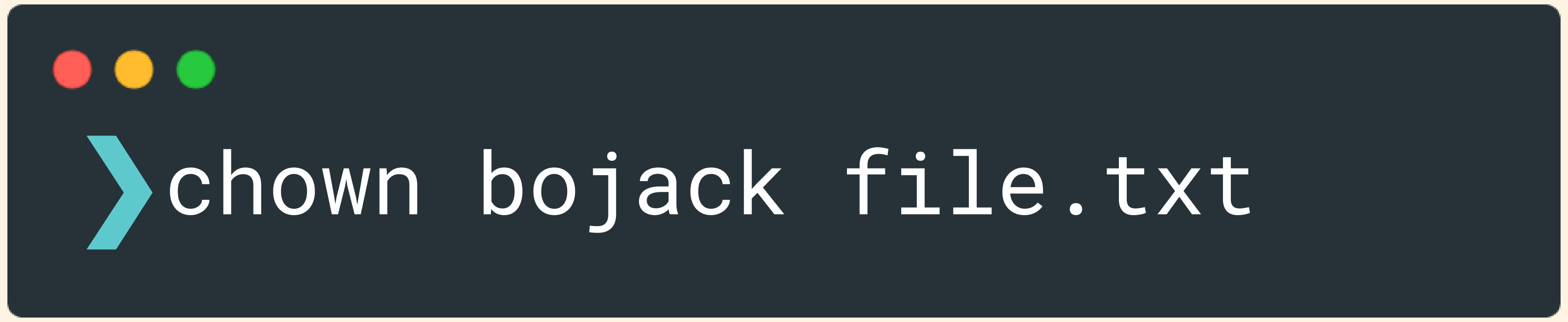
chown

The **chown** command is used to change the owner and/or the group owner of a specific file or directory.

To make bojack the owner of file.txt, we would run **chown bojack file.txt**



```
❯chown USER[:GROUP] FILE(s)
```

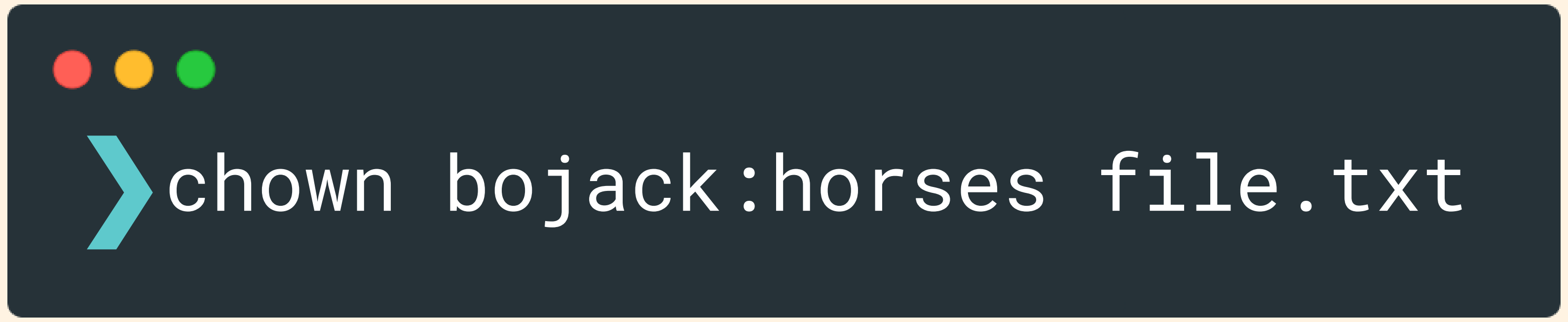


```
❯chown bojack file.txt
```

chown

To change the owner of a file and the file group owner at once we can provide both using **chown USER:GROUP FILE**.

For example, **chown bojack:horses file.txt** will change the owner of file.txt to bojack AND changes the file group owner to the group named horses.

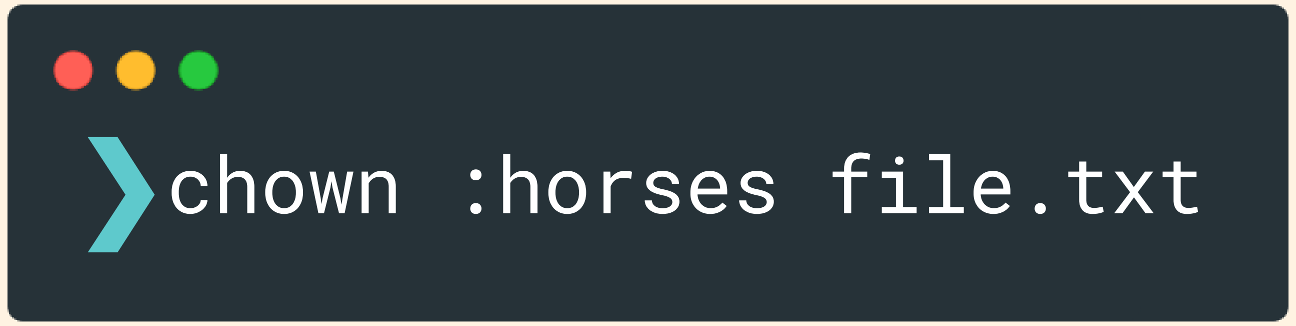


```
❯chown bojack:horses file.txt
```

chown

To only change the group owner of a file, we can run **chown :GROUP FILE**

For example, **chown :horses file.txt** will change the file group owner of file.txt to the group named horses.

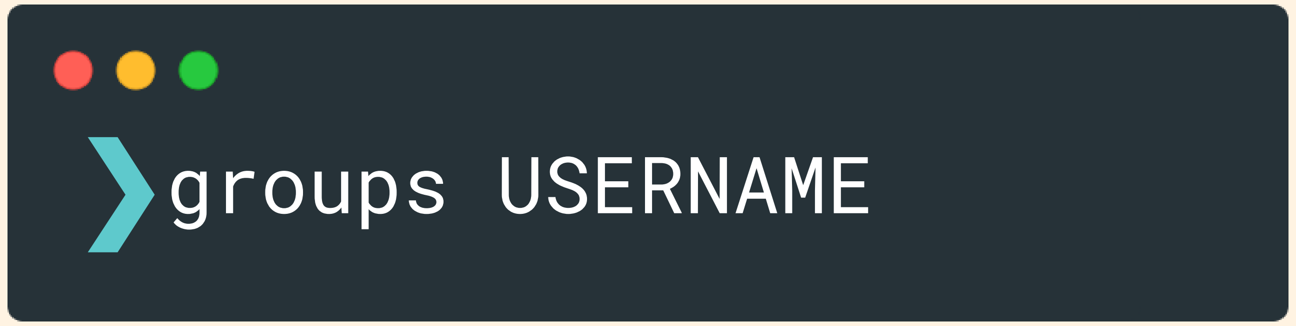


```
chown :horses file.txt
```

groups

To view the groups that a given user belongs to, run **groups USERNAME**

For example, to see which groups hermione is part of, run **groups hermione**

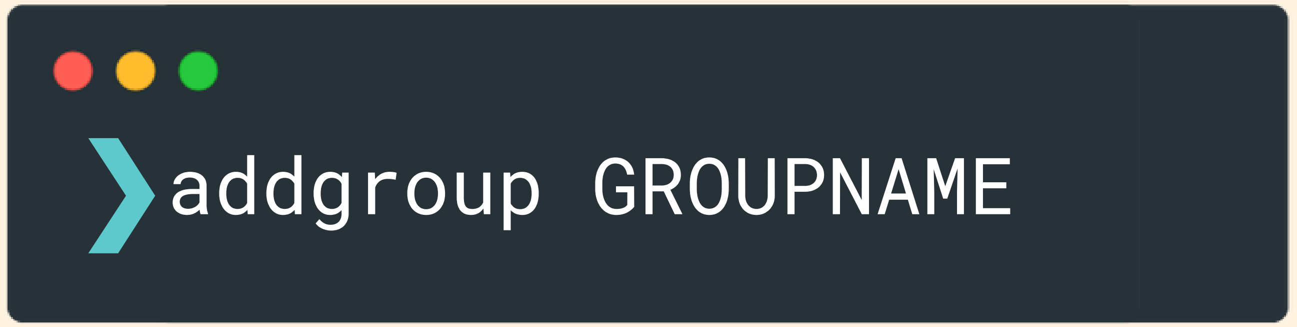


```
groups USERNAME
```

Creating Groups

We can create new groups using the addgroup command.

To create a new group called friends, we would run **addgroup friends**

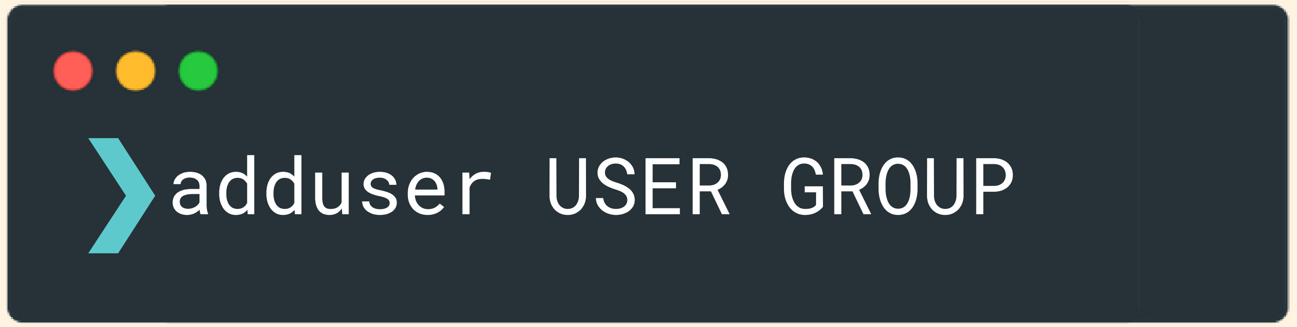


```
addgroup GROUPNAME
```

Adding Group Members

To add a user to a group, use the **adduser** command.

To add hermione to friends, we would run **adduser hermione friends**



```
❯adduser USER GROUP
```

Don't screw around with groups unless you know what you are doing!
This is just a 30 second intro.