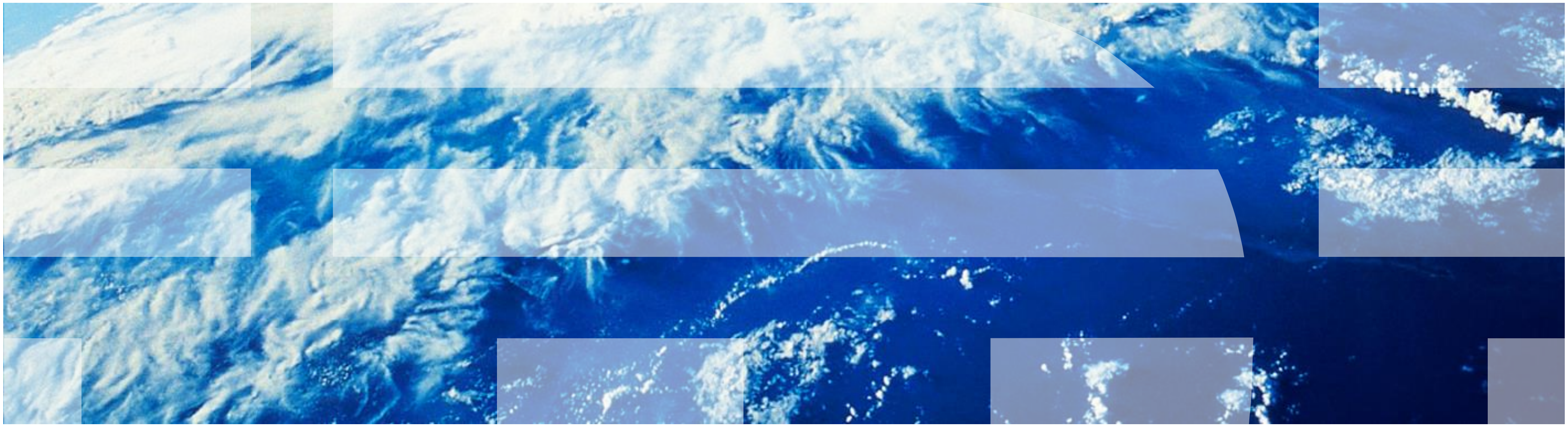
File permissions



Unit objectives

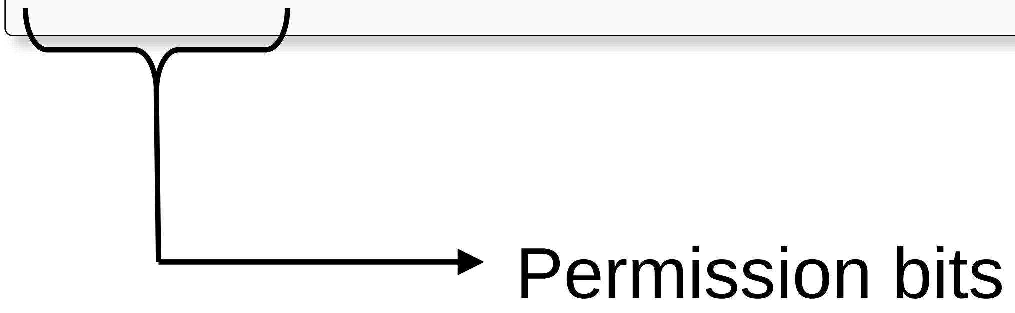
After completing this unit, you should be able to:

- List the basic file permissions
- Change the basic file permissions by using both the octal and symbolic formats
- Use the `umask` command

Long listing of files

- The `ls` command with the `–l` option can be used to obtain more information about the files in a directory.

```
$ ls -l
total 40
drwxr-xr-x    2 team01   staff              256 Apr 29 13:01 c
drwxr-xr-x    2 team01   staff              256 Apr 29 13:13 doc
-rw-r--r--    1 team01   staff            13886 Apr 29 13:02 manuals
-rw-r--r--    1 team01   staff             3331 Apr 29 13:02 test1
```



Permission bits

File protection/permissions

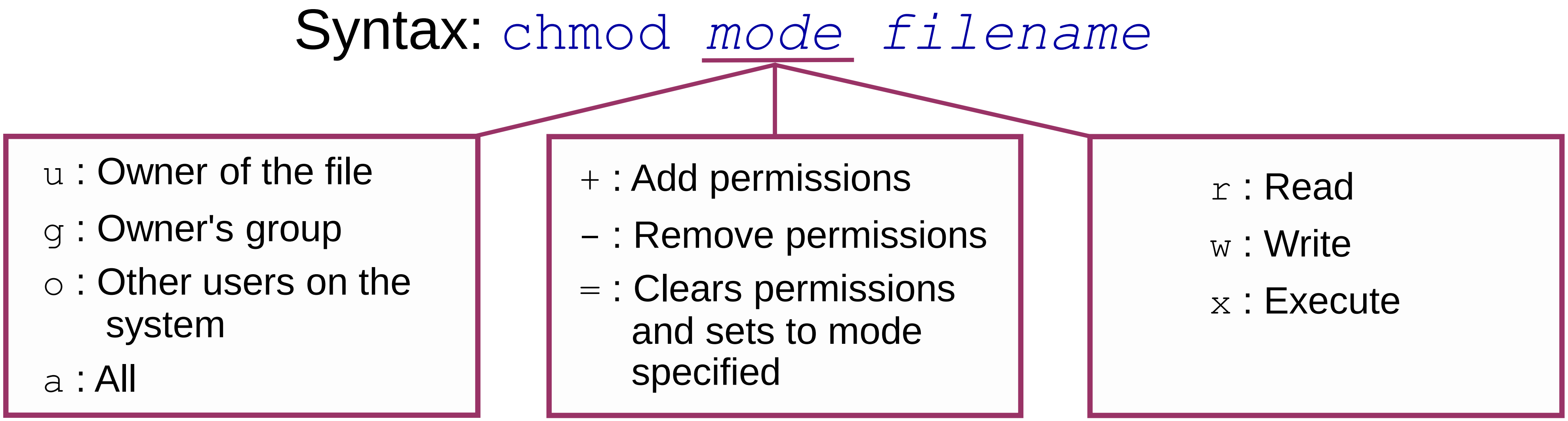
- Every file and directory on the system has file permissions that are associated with it
- Three permission categories: owner, group, and other
- Three bits can be set for each category: read, write, execute (`rwx`)
- For an *ordinary* file:

`r`	Can look at the contents of a file
`w`	Can change or delete the contents of a file
`x`	Can execute the file (r is also needed if a script)

- For a *directory*:

`r`	Can list the files within a directory (`ls`)
`w`	Can modify/remove files in the directory
`x`	Can change into the directory and access the files within (`cd`)

Changing permissions (symbolic notation)



Syntax: `chmod` *`mode`* *`filename`*

- `u` : Owner of the file
- `g` : Owner's group
- `o` : Other users on the system
- `a` : All

- `+` : Add permissions
- `–` : Remove permissions
- `=` : Clears permissions and sets to mode specified

- `r` : Read
- `w` : Write
- `x` : Execute

```
$ ls -l newfile
-rw-r--r-- 1   team01      staff  58 Apr 29        16:06    newfile

$ chmod go+w newfile
$ ls -l newfile
-rw-rw-rw- 1   team01      staff  58 Apr 29        16:06    newfile

$ chmod a+x newfile
$ ls -l newfile
-rwxrwxrwx 1   team01      staff  58 Apr 29        16:06    newfile

$ chmod o-rwx newfile
$ ls -l newfile
-rwxrwx--- 1   team01      staff  58 Apr 29        16:06    newfile
```

Changing permissions (octal notation)

- File and directory permissions can be specified in the symbolic syntax or as an octal number

	User	Group	Others
Symbolic	`rwx`	`rw-`	`r--`
Binary	111	110	100
Conversion	4+2+1	4+2	4
Octal	7	6	4

- Change permissions so the owner and group have read and write permissions and others read only:

```
$ ls -l newfile
-rw-r--r--  1  team01  staff  58 Apr 29     16:06  newfile

$ chmod 664 newfile
$ ls -l newfile
-rw-rw-r--  1  team01  staff  65 Apr 29     17:06  newfile
```

The umask command

- Determines the permissions for new files and directories
- Display and set using the umask command
- The default umask value is 022 and is set in /etc/security/user
  - It can be changed for all users or for a specific user
- The final permission is the complement of the umask value:

	Directory	File	Directory	File
System default	777	666	777	666
Subtract umask	022	022	027	027
Assigned permissions	755	644	750	640

```
$ touch new.file
$ mkdir new.dir
$ umask 027
$ touch new027.file
$ mkdir new027.dir
$ ls -l
total 0
drwxr-xr-x    2 team01   staff          256 Apr 29 20:32 new.dir
-rw-r--r--    1 team01   staff            0 Apr 29 20:32 new.file
drwxr-x---    2 team01   staff          256 Apr 29 20:33 new027.dir
-rw-r-----    1 team01   staff            0 Apr 29 20:33 new027.file
```

Activity: Personal directories

Function/permissions required

Command	Source Directory	Source File	Target Directory
`cd`	x	N/A	N/A
`ls`	r	N/A	N/A
`ls -l`	r, x	N/A	N/A
`mkdir`	x w (parent)	N/A	N/A
`rmdir`	x w (parent)	N/A	N/A
`cat, pg, more`	x	r	N/A
`mv`	x, w	NONE	x, w
`cp`	x	r	x, w
`touch`	x, w *	NONE	N/A
`rm`	x, w	NONE	N/A

Exercise: File permissions

Unit summary

Having completed this unit, you should be able to:

- List the basic file permissions
- Change the basic file permissions by using both the octal and symbolic formats
- Use the `umask` command