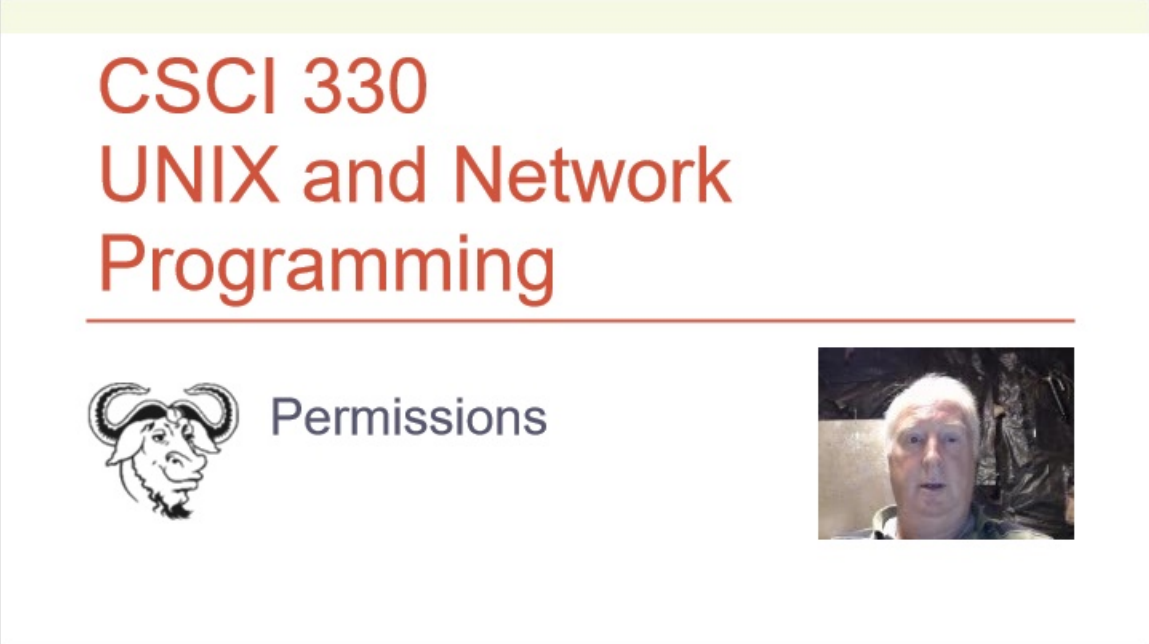
Permissions

1. Permissions

1.1 CSCI 330



1.2 Permissions

Permissions

- all access to directories and files is controlled
- UNIX uses discretionary access control (DAC) model
  - each directory/file has owner
  - owner has discretion over access control details
- access control includes
  - read, write: to protect information
  - execute: to protect state of system
- exception: super user

1.3 User Terminology

User Terminology

- user
  - any one who has account on the system, listed in /etc/passwd
  - protected via password, listed in /etc/shadow
  - internally recognized via a number called “user id”
- group
  - users are organized into groups, listed in /etc/group
  - user can belong to multiple groups
- super user, root
  - has user id “0”
  - responsible for system administration

1.4 File/Directory access

File/Directory access

- file or directory has owner, i.e. the user who created it
- owner sets access permissions
  - access mode: read, write, execute
  - accessor category: self, group, others
- ownership change via: **`chown`**

1.5 Access Modes

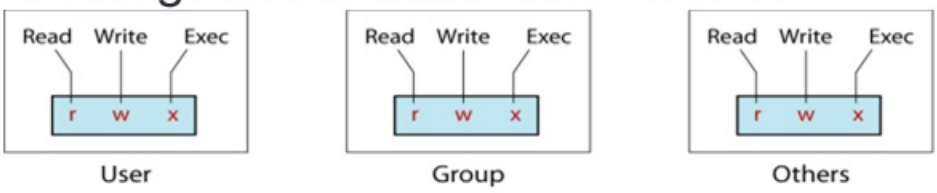
Access Modes

	Meaning on File	Meaning on Directory
r (read)	View file contents (open, read)	List directory contents
w (write)	Change file contents	Change directory contents
x (execute)	Run executable file	Make it current directory, search for files in it

1.6 Accessor Categories

Accessor Categories

- 3 categories of users want access



1.7 Checking Permissions

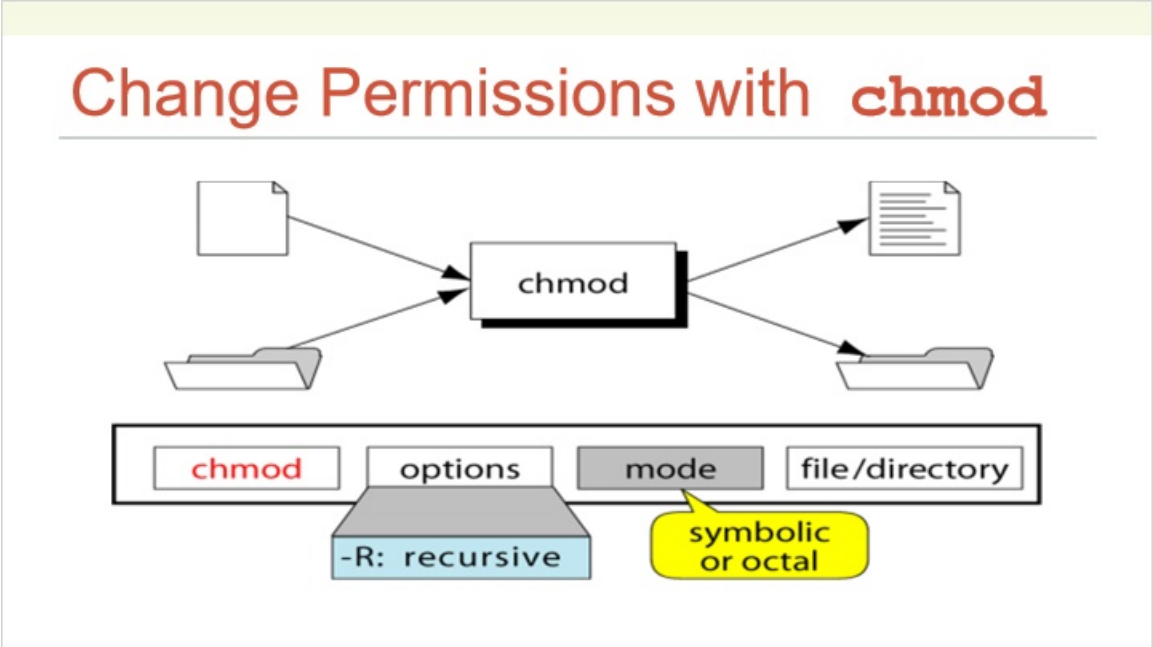
Checking Permissions

- To check the permissions of an existing file or an existing directory, use the “ls -l” command:

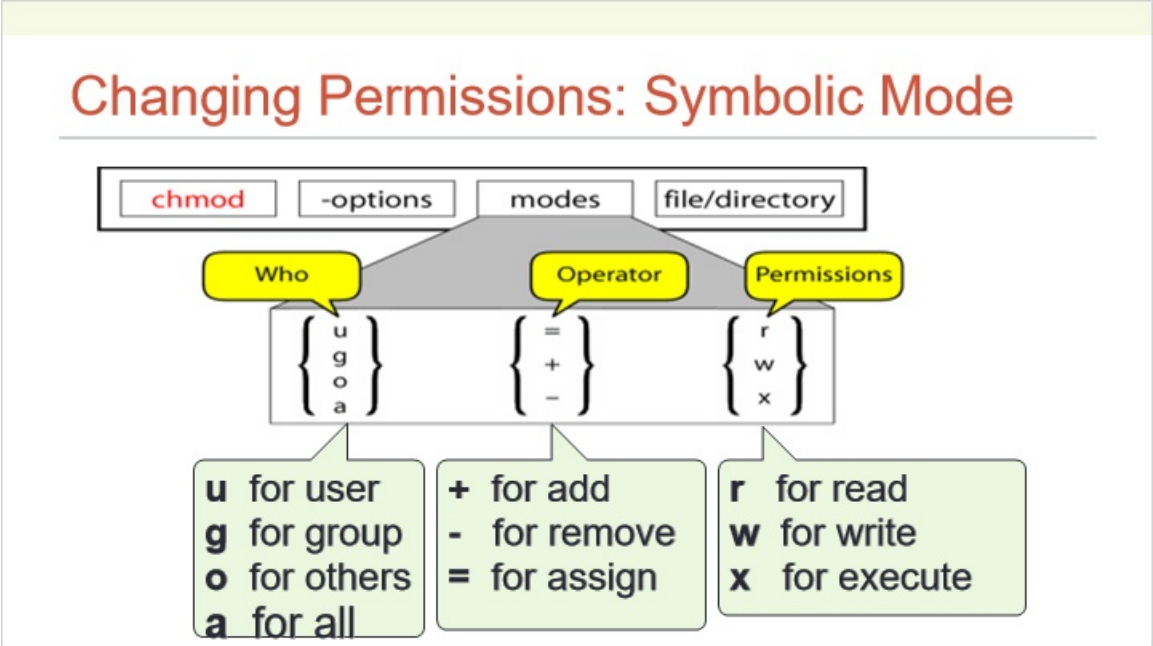
Example:

```
% ls -l
drwx------ 1 z036473  student  86  Feb  7 19:22  scripts
-rw-rw-r-- 1 z036473  student  20  Feb  9 11:25  out.txt
-rwxr-xr-- 1 z036473  student  34  Feb  3 19:42  checkIt
-rw-r--r-- 1 z036473  student  34  Feb  5  9:05  a2.png
```

1.8 Change Permissions with chmod



1.9 Changing Permissions: Symbolic Mode



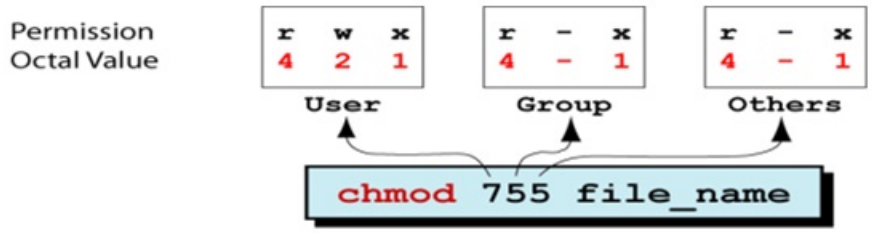
1.10 Examples: Symbolic Mode

Examples: Symbolic Mode

```
% chmod u-w file.txt
% chmod u+w file.txt
% chmod u+x script.sh

% chmod g-w file.txt
% chmod o-rw file.txt

% chmod ug=rwx play.cc
% chmod a+wx other.html

% chmod u+x,go=r script.sh
```

1.11 Changing Permissions: Octal Mode

Changing Permissions: Octal Mode



1.12 Changing Permissions: Octal Mode

Changing Permissions: Octal Mode

Step	Perform…	Settings
1	List the desired setting	rwx|r-x|r-x
2	Assign binary: 1 for access; 0 for no access	111|101|101
3	List octal values for the corresponding binary 1's	421|401|401
4	Convert the octal values to a 3-digit number	7 | 5 | 5
5	Write the command	chmod 755 sort.c

```
% ls -l sort.c
-rwxr-xr-x 1 ege  csci 80 Feb 27 12:23 sort.c
```

1.13 Changing Permissions: example

Changing Permissions: example

- Goal: set mode of file "myfile"
  - Read, write, and execute permissions to self/owner
  - Read and execute permissions to group
  - Execute only permission to others
- We want: `rwx r-x --x`

  Symbolic Mode: `chmod u=rwx,g=rx,o=x myfile`

  Octal Mode: `chmod 751 myfile`

1.14 Special Permissions

Special Permissions

- The regular file permissions (rwx) are used to assign security to files and directories
- 3 additional special permissions can be optionally used on files and directories
  - Set User Id (SUID)
  - Set Group ID (SGID)
  - Sticky bit

1.15 Special Permissions: SUID

Special Permissions: SUID

- SUID used for executable files
  - makes executable run with privileges of file owner, rather than invoker
- Example:
  - "passwd" command and file "/usr/bin/passwd"

```
-rwsr-xr-x 1 root root 41284 Apr 8 21:40 /usr/bin/passwd
```

- allows regular user access to otherwise protected system files while changing password

1.16 Special Permissions: SGID

Special Permissions: SGID

- for executable files
  - logic is similar to SUID bit
  - runs program with group permission of file, rather than group of invoker
- for directories
  - a file created in the directory will be owned by the group owner of the directory, not the group of the user that created the file

1.17 Special Permissions: Sticky Bit

Special Permissions: Sticky Bit

- for executable files:
  - executable is kept in memory even after it ended (no longer used, since modern virtual memory methods are more advanced)
- for directories:
  - file can only be deleted by the user that created it

1.18 Special Permissions: display

Special Permissions: display

- "ls -l" command does not have a section for special permission bits
- however, since special permissions required "execute", they mask the execute permission when displayed using the "ls -l" command

r w x r w x r w x

r w s r w s r w t

SUID SGID STICKY BIT

1.19 Setting Special Permissions

Setting Special Permissions

suid	sgid	stb	r	w	x	r	w	x	r	w	x
4	2	1	4	2	1	4	2	1	4	2	1
7 Special			7 user			7 group			7 others		

Use the "chmod" command with octal mode:

- chmod 7777 filename

1.20 Setting Special Permissions

Setting Special Permissions

- chmod with symbolic notation:

u+s	add SUID
u-s	remove SUID
g+s	add SGID
g-s	remove SGID
+s	add SUID and SGID
+t	set sticky bit

1.21 File mode creation mask

File mode creation mask

- umask (user mask)
  - governs default permission for files and directories
  - sequence of 9 bits: 3 times 3 bits of rwx
  - default: 000 000 010 (002)
    000 010 010 (022) on turing/hopper
- in octal form its bits are removed from:
  - for a file: 110 110 110 (666)
  - for a directory: 111 111 111 (777)
- permission for new
  - file: 110 110 100 (664)
  - directory: 111 111 101 (775)

1.22 User Mask value examples

User Mask value examples

	Directory Default: 777	File Default: 666
000	777 (rwx rwx rwx)	666 (rw- rw- rw-)
111	666 (rw- rw- rw-)	666 (rw- rw- rw-)
222	555 (r-x r-x r-x)	444 (r-- r-- r--)
022	755 (rwx r-x r-x)	644 (rw- r-- r--)
002	775 (rwx rwx r-x)	664 (rw- rw- r--)
066	711 (rwx --x --x)	600 (rw- --- ---)
666	111 (--x --x --x)	000 (--- --- ---)
777	000 (--- --- ---)	000 (--- --- ---)

default

default

1.23 Change the permission default

Change the permission default

- command to display: umask
  - uses a leading zero: 0002 or 0022
- umask -S
  - u=rwx,g=rx,o=rx

- command to change: umask
  - tolerates leading zero
  - ex:

```
% umask 0077
% umask a-r
```

1.24 Summary

Summary

- r, w, x
  - and extra bits (s,t)
- user (self, owner), group, others
- file mode creation mask: umask