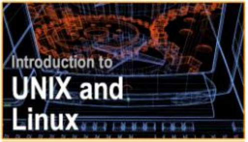


CST 126 – LESSON 10

File Security, Setting and Using Permissions

Chapter 9

Objectives

- To show the three protection and security mechanisms that UNIX provides
- To describe the types of users of a UNIX file
- To discuss the basic operations that can be performed on a UNIX file
- To explain the concept of file access permissions/ privileges in UNIX
- To discuss how a user can determine access privileges for a file
- To describe how a user can set and change permissions for a file
- To cover the commands and primitives
  - ? , ~ , * , chmod, groups, ls – l, ls – ld, umask

Password-based Protection

- All login names are public knowledge and can be found in the /etc/passwd file.
- Change password using:
  - `yppasswd, nispasswd`
- Discovering a user's password:
  - 1) You, as the owner of an account, inform others of your password
  - 2) a password can be guessed by another user
  - 3) a user's password can be extracted by "brute force"

Encryption-based Protection

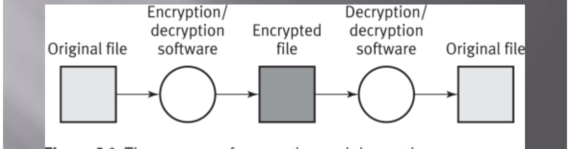
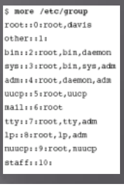

Original file
Encryption/ decryption software
Encrypted file
Decryption/ decryption software
Original file

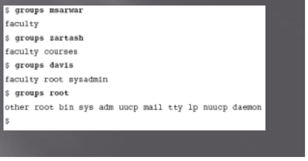
Figure 8.1 The process of encryption and decryption

Protection based on Access Permission

- Types of users
  - User (owner), group, others
  - A user with multiple groups
- Types of Access Permissions
  - Read, write, and execute
- Access Permissions for Directories
  - Directory search



```
$ more /etc/group
root::0:root,davis
other::1:
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm
adm::4:root,daemon,adm
uucp::5:root,uucp
mail::6:root
tty::7:root,tty,adm
lp::8:root,lp,adm
nuucp::9:root,nuucp
staff::10:
```



```
$ groups msarwar
faculty
$ groups zartash
faculty courses
$ groups davis
faculty root sysadmin
$ groups root
other root bin sys adm uucp mail tty lp nuucp daemon
$
```

Protection based on Access Permission (Contd)

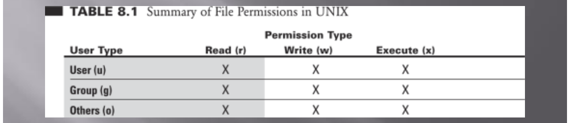
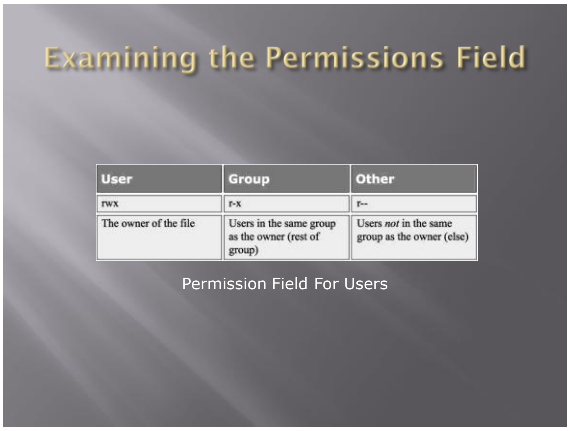


TABLE 8.1 Summary of File Permissions in UNIX

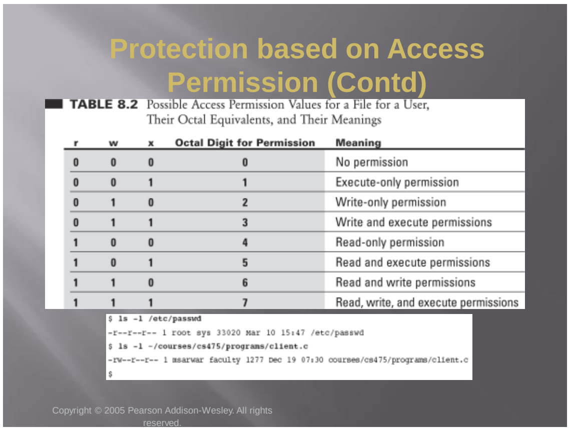
	Permission Type		
User Type	Read (r)	Write (w)	Execute (x)
User (u)	X	X	X
Group (g)	X	X	X
Others (o)	X	X	X



Examining the Permissions Field

User	Group	Other
rwx	r-x	r--
The owner of the file	Users in the same group as the owner (rest of group)	Users *not* in the same group as the owner (else)

Permission Field For Users

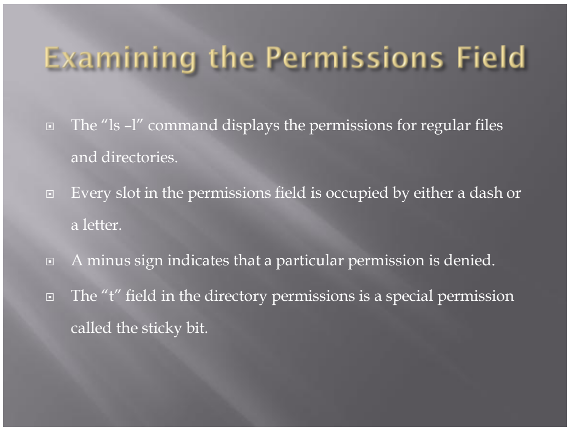


Protection based on Access Permission (Contd)

TABLE 8.2 Possible Access Permission Values for a File for a User, Their Octal Equivalents, and Their Meanings

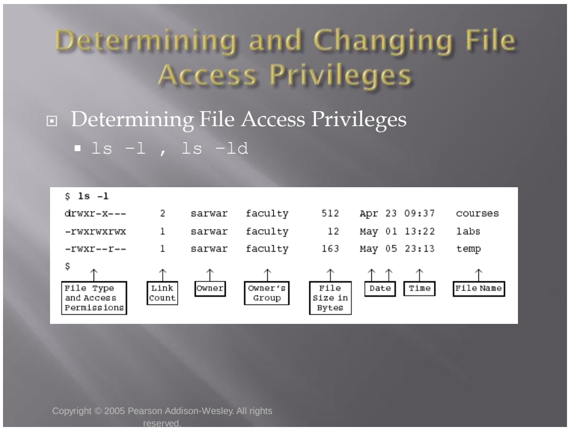
r	w	x	Octal Digit for Permission	Meaning
0	0	0	0	No permission
0	0	1	1	Execute-only permission
0	1	0	2	Write-only permission
0	1	1	3	Write and execute permissions
1	0	0	4	Read-only permission
1	0	1	5	Read and execute permissions
1	1	0	6	Read and write permissions
1	1	1	7	Read, write, and execute permissions

```
$ ls -l /etc/passwd
-r--r--r-- 1 root sys 33020 Mar 10 15:47 /etc/passwd
$ ls -l ~/courses/cs475/programs/client.c
-rw--r--r-- 1 msarwar faculty 1277 Dec 19 07:30 courses/cs475/programs/client.c
$
```



Examining the Permissions Field

- The "ls –l" command displays the permissions for regular files and directories.
- Every slot in the permissions field is occupied by either a dash or a letter.
- A minus sign indicates that a particular permission is denied.
- The "t" field in the directory permissions is a special permission called the sticky bit.



Determining and Changing File Access Privileges

- Determining File Access Privileges
  - `ls -l , ls -ld`

```
$ ls -l
drwxr-x---    2   sarwar   faculty    512   Apr 23 09:37   courses
-rwxrwxrwx    1   sarwar   faculty     12   May 01 13:22   labs
-rwxr--r--    1   sarwar   faculty    163   May 05 23:13   temp
$
```

File Type and Access Permissions

Link Count

Owner

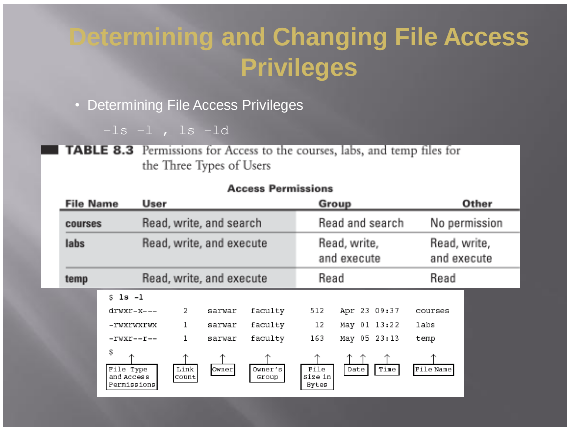
Owner's Group

File Size in Bytes

Date

Time

File Name



Determining and Changing File Access Privileges

- Determining File Access Privileges

  -ls -l , ls -ld

TABLE 8.3 Permissions for Access to the courses, labs, and temp files for the Three Types of Users

	Access Permissions		
File Name	**User**	**Group**	**Other**
courses	Read, write, and search	Read and search	No permission
labs	Read, write, and execute	Read, write, and execute	Read, write, and execute
temp	Read, write, and execute	Read	Read

```
$ ls -l
drwxr-x---    2   sarwar   faculty    512   Apr 23 09:37   courses
-rwxrwxrwx    1   sarwar   faculty     12   May 01 13:22   labs
-rwxr--r--    1   sarwar   faculty    163   May 05 23:13   temp
$
```

File Type and Access Permissions

Link Count

Owner

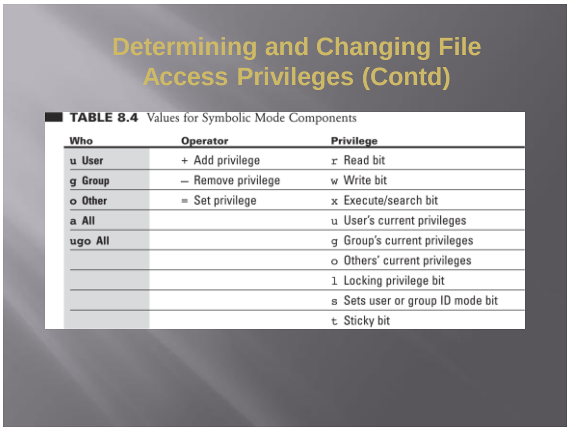
Owner's Group

File Size in Bytes

Date

Time

File Name



Determining and Changing File Access Privileges (Contd)

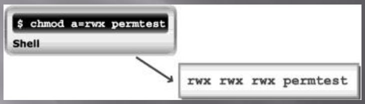
TABLE 8.4 Values for Symbolic Mode Components

Who	Operator	Privilege
u User	+ Add privilege	r Read bit
g Group	– Remove privilege	w Write bit
o Other	= Set privilege	x Execute/search bit
a All		u User's current privileges
ugo All		g Group's current privileges
		o Others' current privileges
		l Locking privilege bit
		s Sets user or group ID mode bit
		t Sticky bit

Changing File Permissions Using Mnemonics

- The "chmod" command can accept permission settings in the form of letter arguments or numbers.
- The mnemonic assignment method allows a user to set permissions for each type of user in several ways.
- Assigning specific permissions.
- Adding and deleting permissions.

Assigning Specific Permissions

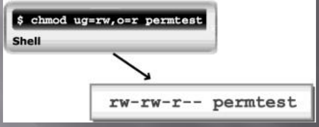


```
$ chmod a=rwx permtest
```

```
rwx rwx rwx permtest
```

Assigning All Permissions to All Users

Assigning Specific Permissions

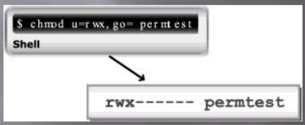


```
$ chmod ug=rw,o=r permtest
```

```
rw-rw-r-- permtest
```

Assigning Specific Permissions to Specific Users

Adding and Deleting Permissions



```
$ chmod u=rwx, go= permtest
```

```
rwx------ permtest
```

Denying Specific Permission to Specific Users

Changing File Permissions Numerically

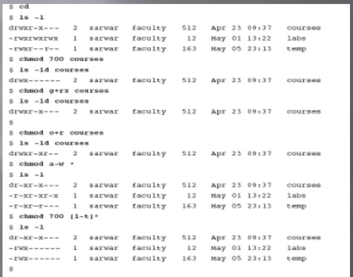
- Numbers can also be used for conveying permissions information for all the three types of users.
- The number 700 specifies the rwx permissions only for the owner of a file.
- The numerical approach allows a user to specify the exact permissions to be granted regardless of the current permission.

Changing File Permissions Numerically

- Combination permissions are specified using the sum of the values for the specific permissions.
- The primitives (0, 1, 2, and 4) can be added to grant any combination of permissions.
- The combination of the three numbers 1, 2, and 4 can be used to express the eight possible combinations of execute, write, and read permissions.

Determining and Changing File Access Privileges

- Changing File Access Privileges
  - `chmod [options] octal-mode file-list`
  - `chmod [options] symbolic-mode file-list`



```
$ cd
$ ls -l
drwxr-x---   2  sarwar  faculty  512  Apr 23 09:37  courses
-rwxrwxrwx   1  sarwar  faculty   12  May 01 13:22  labs
-rwxr--r--   1  sarwar  faculty  163  May 05 23:13  temp
$ chmod 700 courses
$ ls -ld courses
drwx------   2  sarwar  faculty  512  Apr 23 09:37  courses
$ chmod g+rx courses
$ ls -ld courses
drwxr-x---   2  sarwar  faculty  512  Apr 23 09:37  courses
$
$ chmod o+r courses
$ ls -ld courses
drwxr-xr--   2  sarwar  faculty  512  Apr 23 09:37  courses
$ chmod a-w *
$ ls -l
dr-xr-x---   2  sarwar  faculty  512  Apr 23 09:37  courses
-r-xr-xr-x   1  sarwar  faculty   12  May 01 13:22  labs
-r-xr-r---   1  sarwar  faculty  163  May 05 23:13  temp
$ chmod 700 [l-t]*
$ ls -l
dr-xr-x---   2  sarwar  faculty  512  Apr 23 09:37  courses
-rwx------   1  sarwar  faculty   12  May 01 13:22  labs
-rwx------   1  sarwar  faculty  163  May 05 23:13  temp
$
```

Examples of chmod Command

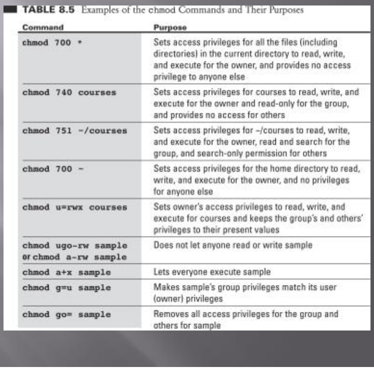
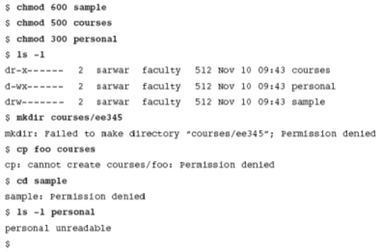


TABLE 8.5 Examples of the chmod Commands and Their Purposes

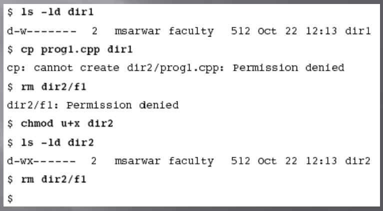
Command	Purpose
`chmod 700 *`	Sets access privileges for all the files (including directories) in the current directory to read, write, and execute for the owner, and provides no access privilege to anyone else
`chmod 740 courses`	Sets access privileges for courses to read, write, and execute for the owner and read-only for the group, and provides no access for others
`chmod 751 ~/courses`	Sets access privileges for ~/courses to read, write, and execute for the owner, read and search for the group, and search-only permission for others
`chmod 700 ~`	Sets access privileges for the home directory to read, write, and execute for the owner, and no privileges for anyone else
`chmod u=rwx courses`	Sets owner's access privileges to read, write, and execute for courses and keeps the group's and others' privileges to their present values
`chmod ugo-rw sample` or `chmod a-rw sample`	Does not let anyone read or write sample
`chmod a+x sample`	Lets everyone execute sample
`chmod g=u sample`	Makes sample's group privileges match its user (owner) privileges
`chmod go= sample`	Removes all access privileges for the group and others for sample

Access Privileges for Directories



```
$ chmod 600 sample
$ chmod 500 courses
$ chmod 300 personal
$ ls -l
dr-x------  2  sarwar  faculty  512 Nov 10 09:43 courses
d-wx------  2  sarwar  faculty  512 Nov 10 09:43 personal
drw-------  2  sarwar  faculty  512 Nov 10 09:43 sample
$ mkdir courses/ee345
mkdir: Failed to make directory "courses/ee345"; Permission denied
$ cp foo courses
cp: cannot create courses/foo: Permission denied
$ cd sample
sample: Permission denied
$ ls -l personal
personal unreadable
$
```

Access Privileges for Directories

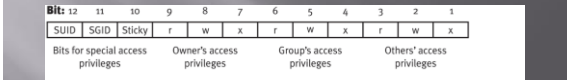


```
$ ls -ld dir1
d-w-------  2   msarwar faculty   512 Oct  22  12:13 dir1
$ cp prog1.cpp dir1
cp: cannot create dir2/prog1.cpp: Permission denied
$ rm dir2/f1
dir2/f1: Permission denied
$ chmod u+x dir2
$ ls -ld dir2
d-wx------  2   msarwar faculty   512 Oct  22  12:13 dir2
$ rm dir2/f1
$
```

Special Access Bits

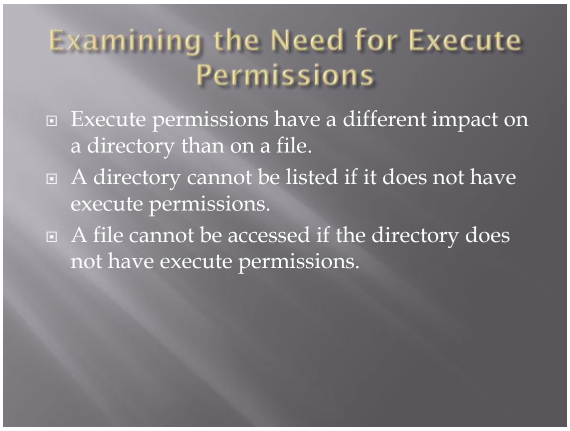
- The Set-User-ID (SUID) Bit
  - If this bit is set for a file containing an executable program for a command, the command takes on the privileges of the owner of the file when it executes.
  - `chmod 4xxx file-list`
  - `chmod u+s file-list`
- The Set-Group-ID (SGID) Bit
  - Causes the access permission of the process to take the group identity of the group to which the owner of the file belongs.
  - `chmod 2xxx file-list`
  - `chmod g+s file-list`
- The Sticky Bit
  - Can be set for a directory to ensure that an unprivileged user cannot remove or rename files of other users in that directory.
  - `chmod 1xxx file-list`
  - `chmod +t file-list`

Special Access Bits



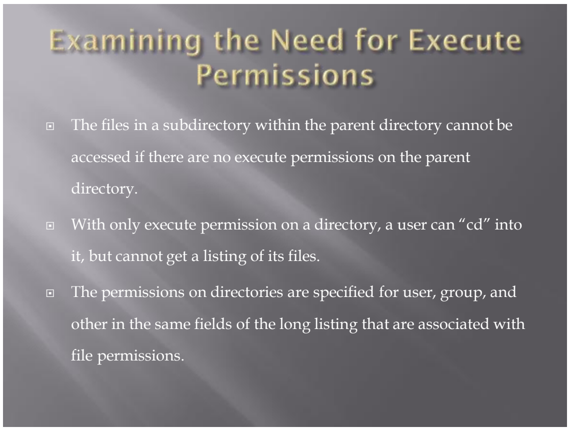
Bit: 12	11	10	9	8	7	6	5	4	3	2	1
SUID	SGID	Sticky	r	w	x	r	w	x	r	w	x
Bits for special access privileges			Owner's access privileges			Group's access privileges			Others' access privileges		

Figure 8.2 Position of access privilege bits for UNIX files as specified in the chmod command



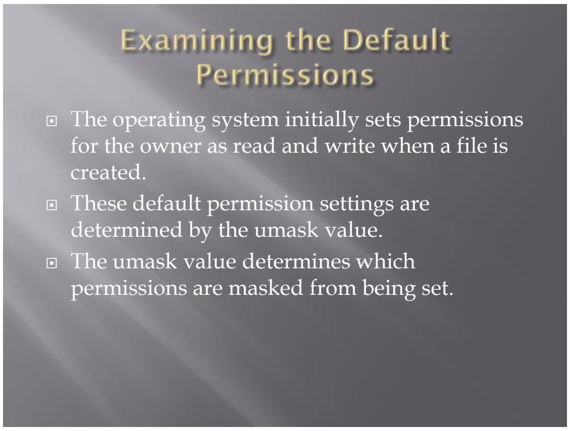
Examining the Need for Execute Permissions

- Execute permissions have a different impact on a directory than on a file.
- A directory cannot be listed if it does not have execute permissions.
- A file cannot be accessed if the directory does not have execute permissions.



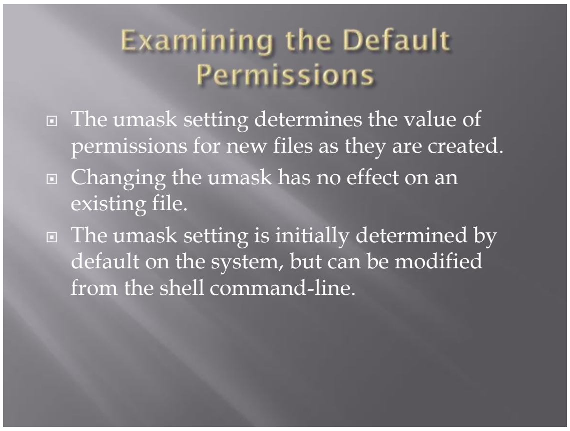
Examining the Need for Execute Permissions

- The files in a subdirectory within the parent directory cannot be accessed if there are no execute permissions on the parent directory.
- With only execute permission on a directory, a user can "cd" into it, but cannot get a listing of its files.
- The permissions on directories are specified for user, group, and other in the same fields of the long listing that are associated with file permissions.



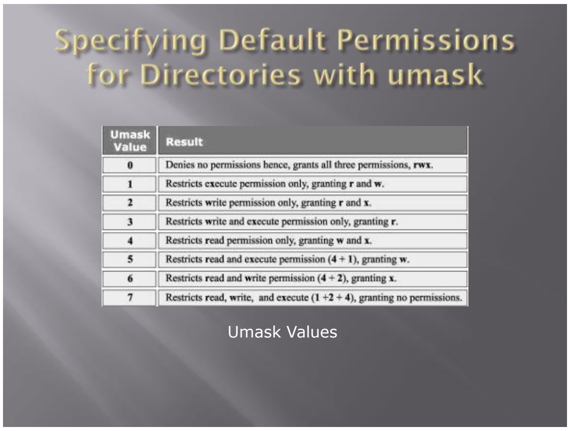
Examining the Default Permissions

- The operating system initially sets permissions for the owner as read and write when a file is created.
- These default permission settings are determined by the umask value.
- The umask value determines which permissions are masked from being set.



Examining the Default Permissions

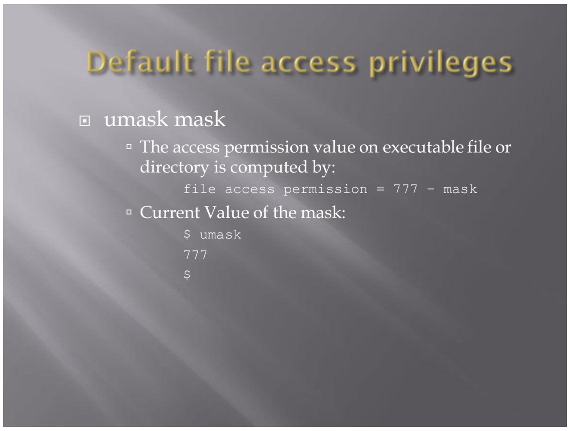
- The umask setting determines the value of permissions for new files as they are created.
- Changing the umask has no effect on an existing file.
- The umask setting is initially determined by default on the system, but can be modified from the shell command-line.



Specifying Default Permissions for Directories with umask

Umask Value	Result
0	Denies no permissions hence, grants all three permissions, **rwx**.
1	Restricts execute permission only, granting **r** and **w**.
2	Restricts write permission only, granting **r** and **x**.
3	Restricts write and execute permission only, granting **r**.
4	Restricts read permission only, granting **w** and **x**.
5	Restricts read and execute permission (4 + 1), granting **w**.
6	Restricts read and write permission (4 + 2), granting **x**.
7	Restricts read, write, and execute (1 +2 + 4), granting no permissions.

Umask Values



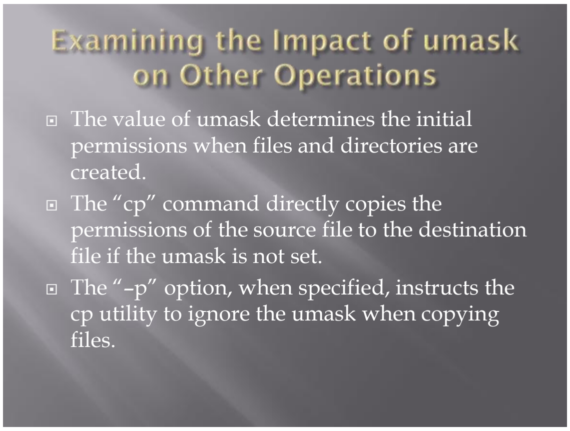
Default file access privileges

- umask mask
  - The access permission value on executable file or directory is computed by:

```
file access permission = 777 – mask
```

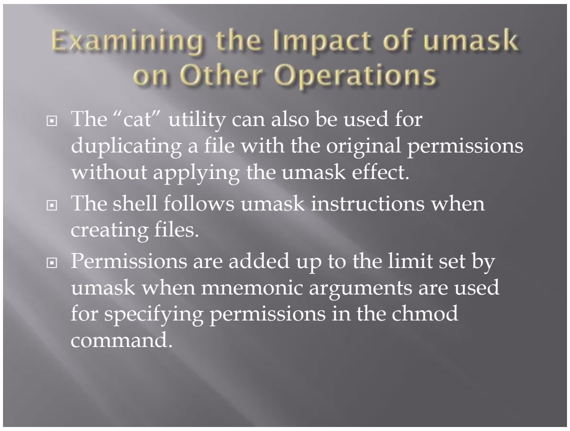
  - Current Value of the mask:

```
$ umask
777
$
```



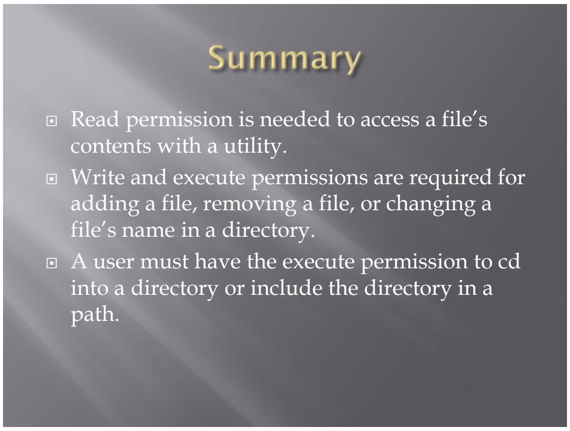
Examining the Impact of umask on Other Operations

- The value of umask determines the initial permissions when files and directories are created.
- The "cp" command directly copies the permissions of the source file to the destination file if the umask is not set.
- The "–p" option, when specified, instructs the cp utility to ignore the umask when copying files.



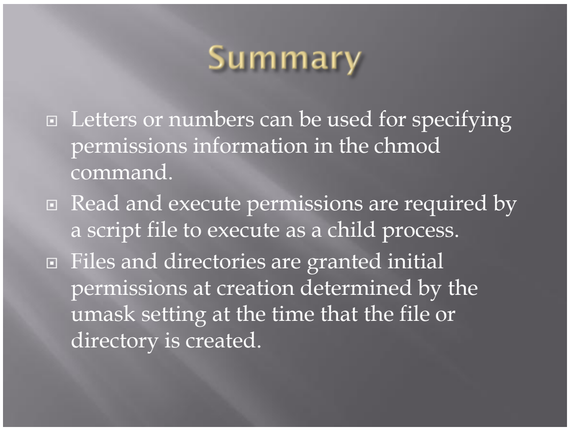
Examining the Impact of umask on Other Operations

- The "cat" utility can also be used for duplicating a file with the original permissions without applying the umask effect.
- The shell follows umask instructions when creating files.
- Permissions are added up to the limit set by umask when mnemonic arguments are used for specifying permissions in the chmod command.



Summary

- Read permission is needed to access a file's contents with a utility.
- Write and execute permissions are required for adding a file, removing a file, or changing a file's name in a directory.
- A user must have the execute permission to cd into a directory or include the directory in a path.



Summary

- Letters or numbers can be used for specifying permissions information in the chmod command.
- Read and execute permissions are required by a script file to execute as a child process.
- Files and directories are granted initial permissions at creation determined by the umask setting at the time that the file or directory is created.