CSCI 330

UNIX File Permissions

Jon Lehuta

Northern Illinois University

August 17, 2020

UNIX File Permissions - Outline

UNIX File Permissions

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

User Terminology

- User
  - Anyone who has account on the system
  - Originally listed in `/etc/passwd`, now in `/etc/shadow`
  - protected via password
  - internally recognized via an integer called the user id
- Group
  - users are organized into groups
  - listed `/etc/group` (and `/etc/gshadow`)
  - a user can belong to multiple groups
- Super user, `root`
  - has user id 0
  - responsible for system administration

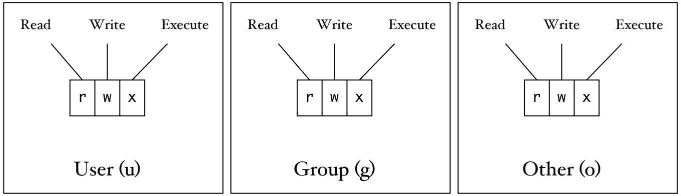
File/Directory access

- Every file/directory has an owner, usually the user who created it.
- The owner sets access permissions
  - access modes: read (`r`), write (`w`), execute (`x`)
  - accessor category: user (owner (`u`)), group (`g`), others (`o`)
- Change ownership via: `chown`

Access Permission Modes

	For File	**For Directory**
`r` (read)	View file contents (open, read)	List directory contents.
`w` (write)	Change file contents	Add/remove files from the directory.
`x` (execute)	Run file as executable.	Able to enter the directory and access files inside. (search)
-	Permission denied.	Permission denied.

Categories of Users

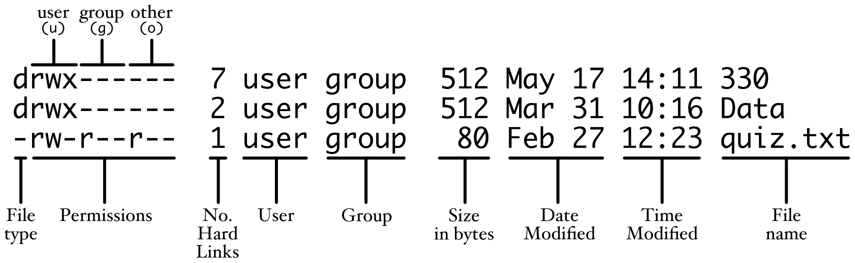


Three categories of user, three types of access for each.

Northern Illinois University

Checking Permissions

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



```
drwx------  7 user group  512 May 17 14:11 330
drwx------  2 user group  512 Mar 31 10:16 Data
-rw-r--r--  1 user group   80 Feb 27 12:23 quiz.txt
```

The permissions of each file are shown with `ls -l`

Change Permissions with chmod

```
% chmod [-options] mode file
```

- mode – octal or symbolic mode for file
- file – name of file to change mode for

Option	Effect
-R	recursively apply to contents of directory

Changing Permissions: Symbolic Mode

- For whom?
  - `u` for user who owns the file
  - `g` for group
  - `o` for others
  - `a` or `ugo` for all three
- Do what?
  - `+` for add
  - `-` for remove
  - `=` for assign/set to
- For what type(s) of access?
  - `r` for read
  - `w` for write
  - `x` for execute/search

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
```

Northern Illinois University

Changing Permissions: Octal Mode

```
 u   g   o  values --> r=4 w=2 x=1
rwx rwx rwx
111 111 111
421 421 421 value for rwx, or 0 if -
  7   7   7 sum of above 3 values
  yields 777

r-- -wx rw-
100 011 110 1 for rwx, 0 for -
400 021 420 value for rwx or 0 if -
  4   3   6 sum of above 3 values
  yield 436
```

Changing Permissions: Octal Mode

Step	Perform	
1	List the desired setting	`rwx|r-x|r-x`
2	Assign binary; 1 for access 0 for none	`111|101|101`
3	List octal values for corresponding 1's	`421|401|401`
4	Add up the numbers for each of u,g,o	`7	5	5`
5	Write up the command	`chmod 755 sort.c`

```
% ls -l sort.c
```

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

Goal: set mode of file `myfile`

- Read, write, and execute permissions to owner
- Read and execute permissions to group
- Execute permission to others

We want:

```
 u   g   o
rwx r-x --x
111 101 001 bits
421 401 001 values
  7   5   1 sum
```

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

Using Octal Mode: `chmod 751 myfile`

Special Permissions

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

Special Permissions: SUID

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

```
-rwsr-xr-x  1  root root 42776 2019-04-04 00:50 passwd
```

    - This allows regular user access to system files while changing password.

Special Permissions: SGID

- logic is similar to SUID bit used for executable files
- runs program with group permission of file, rather than group of invoker
- Example:
  - if a file is owned by the system group and also has the SGID bit set, then if file is executed it runs with system group privileges

Special Permissions: Sticky Bit

- Different uses on different systems, not clearly defined
- 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

Special Permissions: display

`ls -l` command does not have a section for special permission bits

However, since special permissions all required "execute", they mask the execute permission when displayed using the `ls -l` command.

```
r w x  r w x  r w x <- normal
    |      |      |
r w s  r w s  r w t <- special
    ^      ^      ^
 suid   guid sticky
```

Setting Special Permissions

bit values (in order):

suid	sgid	sticky	r	w	x	r	w	x	r	w	x
4	2	1	4	2	1	4	2	1	4	2	1

as octal:

special	user	group	other
7	7	7	7

Use the `chmod` command with octal mode:

```
% chmod 7777 filename
```

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

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: `000 010 010` binary (`022` octal)
- 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: `110 100 100` (`644`)
- directory: `111 101 101` (`755`)

User Mask values

umask	Directory default: `777`	File default: `666`
`000`	`777 rwxrwxrwx`	`666 rw-rw-rw-`
`111`	`666 rw-rw-rw-`	`666 rw-rw-rw-`
`222`	`555 r-xr-xr-x`	`444 r--r--r--`
`333`	`444 r--r--r--`	`444 r--r--r--`
`444`	`333 -wx-wx-wx`	`222 -w--w--w-`
`555`	`222 -w--w--w-`	`222 -w--w--w-`
`666`	`111 --x--x--x`	`000 ---------`
`777`	`000 ---------`	`000 ---------`

Change the permission default

- Command to display: `umask`
  - uses a leading zero, i.e. `0022`
- command to change: umask
  - tolerates leading zero

```
% umask -S u=rwx,g=rx,o=rx
% umask 0077
% umask a-r
```

Summary

r, w, x

- and extra bits (s,t)

user (self, owner), group, others

file mode creation mask: umask