Linux filesystem permissions

David Morgan

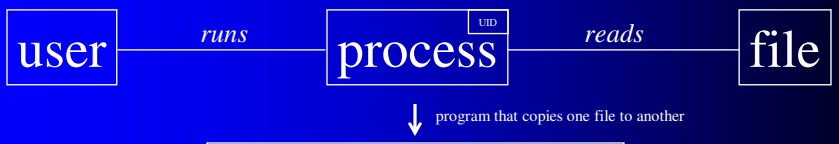
An access control mechanism

- For granting/withholding access to a resource
- Based on relation between file- and user-characteristics
- Analogy
  - government documents receive classifications
  - government employees receive clearances
  - access to particular document by particular employee determined by relation between classification and clearance

Bigger picture - how we think of it

Bigger picture - how it actually works

users don't read files, processes do



program that copies one file to another

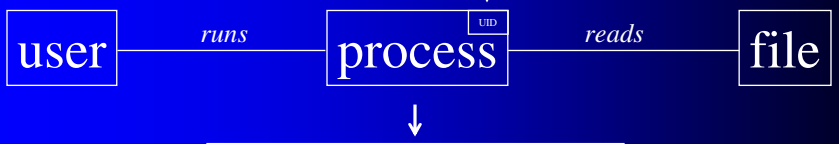
```
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
int main()
{
   char c;
   int in, out;

   in = open("file.in", O_RDONLY);

   out = open("file.out", O_WRONLY|O_CREAT, S_IRUSR|S_IWUSR);

   while(read(in,&c,1) == 1)

      write(out,&c,1);

   exit(0);
}
```

note system calls "open" "read" "write"
They do the file access
user? *isn't even mentioned* in the calls

Bigger picture - how it actually works

AUTHENTICATION HERE
up front, determines account
for first (shell) process

same account, carried forward by inheritance
from shell process to this spawned one




```
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
int main()
{
   char c;
   int in, out;

   in = open("file.in", O_RDONLY);

   out = open("file.out", O_WRONLY|O_CREAT, S_IRUSR|S_IWUSR);

   while(read(in,&c,1) == 1)

     write(out,&c,1);

   exit(0);
}
```

note system calls "open" "read" "write"
They do the file access
user? *isn't even mentioned* in the calls

Government authorization

- documents have "classifications"
- employees have "clearances"
  - confidential
  - secret
  - top secret

$$z = f(x, y)$$

access decision =  = f (document's classification, clearance)

Computer auth not so different

- linux
  - files have permissions for particular user accounts
  - processes (the *true* file "users") carry a user account identity
- Windows
  - resource security policies
  - processes carry user and group affiliation

access decision =  = f (file's permissions, user)

Linux users

- system keeps a list of user accounts
- system usage demands a user identification
  - supplied at login… no login, no usage
- a user id is implicit in all session activities
  - all session activities are performed by processes
  - every process has some user id as an attribute
  - helps determine access to resources by that process
- users can be grouped

Files have (1) a user affiliation

Files' user affiliations are shown by the ls –l command:

```
[root@EMACH1 schools]# ls -l
total 12
-rw-r--r--  1 root     students     121 Dec  8 17:15 assignments
-rw-rw----  1 root     teachers     119 Dec  8 17:13 grades
-rw-r-----  1 root     administ      95 Dec  8 17:10 salaries
```

Files

Their affiliated users

Files have (2) a group affiliation

Files' group affiliations are shown by the ls –l command:

```
[root@EMACH1 schools]# ls -l
total 12
-rw-r--r--  1 root     students     121 Dec  8 17:15 assignments
-rw-rw----  1 root     teachers     119 Dec  8 17:13 grades
-rw-r-----  1 root     administ      95 Dec  8 17:10 salaries
```

Files

Their affiliated groups

Files have (3) a permissions setting

Files' permissions settings are shown by the ls –l command:

```
[root@EMACH1 schools]# ls -l
total 12
-rw-r--r--  1 root    students    121 Dec  8 17:15 assignments
-rw-rw----  1 root    teachers    119 Dec  8 17:13 grades
-rw-r-----  1 root    administ     95 Dec  8 17:10 salaries
```

Files

Their permissions settings

Where? : inode structure of a file

found in inode table of an ext filesystem*

field size	start	end	Item	
2	1	2	File type and access rights	← permissions setting here
2	3	4	Owner identification	← user affiliation here
4	5	8	File length in bytes	
4	9	12	Time of last file access	
4	13	16	Time that inode last changed	
4	17	20	Time that file contents last changed	
4	21	24	Time of file deletion	
2	25	26	Group identifier	← group affiliation here
2	27	28	Hard links counter	
4	29	32	Number of data blocks of the file	
4	33	36	File flags	
4	37	40	Specific operating system information	
4	41	44	Pointer to first data block	
56	45	100	14 more pointers to data blocks	
4	101	104	File version (for NFS)	
4	105	108	File access control list	
4	109	112	Directory access control list	← pointer to additional data of variable length here ("extended attributes") e.g., ACL, SELinux labels
4	113	116	Fragment address	
8	117	124	Specific operating system information	

* this is unique and specific to the ext filesystem family. File utilities (e.g., ls or chmod) often assume it. Applying them to other filesystems (such as USB drives' usual FAT32) can produce unpredictable or unmeaningful results.

Users have group memberships

Users' memberships appear in the file that defines the groups, (/etc/group) not the one that defines the users (/etc/passwd)

file /etc/group

The group

```
.
administrators:x:542:socrates,roy
teachers:x:543:plato
students:x:544:aristotle
.
.
```

The members

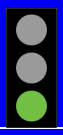
File system - permissions

-rwxr-x---

- **File type** (file, directory, device,…)
- Accesses granted to **file's associated User**
- Accesses granted to members of **file's Group***
- Accesses granted to all **Other users**

*other than the associated user

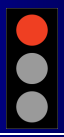
Meaning for files



letter :

- r – can read
  - – can open file
- w – write
  - – can modify file
- x – execute
  - – can try to execute file

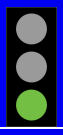
-or else-



hyphen :

- - – can't read
  - – can't open file
- - – can't write
  - – can't modify file
- - – can't execute
  - – can't try to execute file

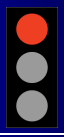
Meaning for directories



letter :

- r – can read
  - – can view contained files
- w – write
  - – can change contained files (add, rename, move)
- x – execute
  - – can enter directory (cd)
  - – can open contained files in directory or its subs

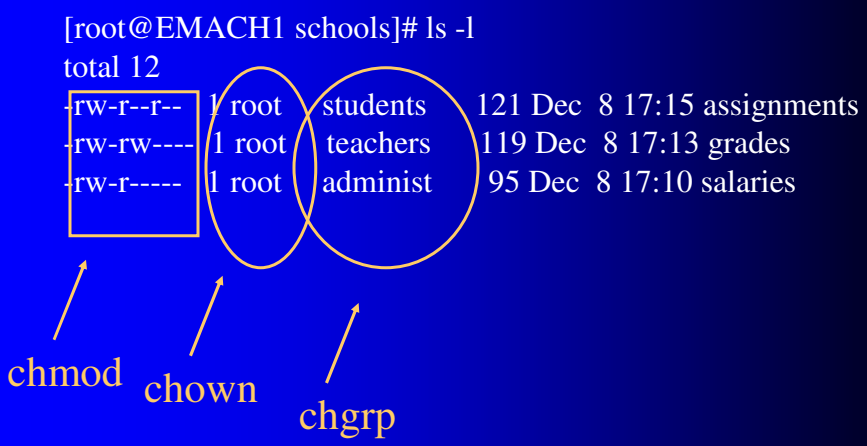
-or else-



hyphen :

- - – can't read
  - – can't view contained files
- - – can't write
  - – can't change contained files (add, rename, move)
- - – can't execute
  - – can't enter directory (cd)
  - – can't open contained files in directory or its subs

Commands for controlling these



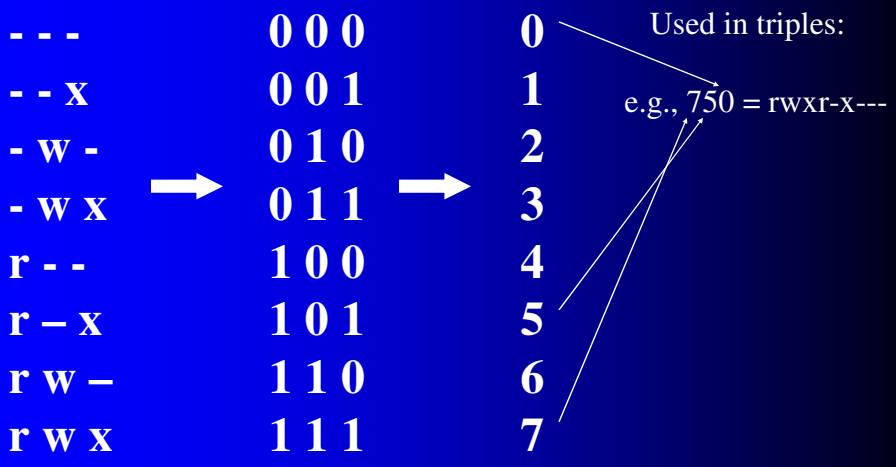
chmod – change file permissions

- To restrict/extend access to others
- To enable script execution

chmod – change file permissions

- "entire" granularity (all 9-at-a-time)
  - use octal specification
- "surgical" granularity (just 1, or a couple, at a time)
  - use who/how/what specification

changing all permissions – octal specification



changing just some permissions – who/how/what specification

who	how	what
u	+	r
g	-	w
o	=	x
a		s

who/how/what

- **u** – for that user associated with the file (“owner”)
- **g** – for those users in group associated with the file
- **o** – for anybody else (“world”)
- a – all three of them

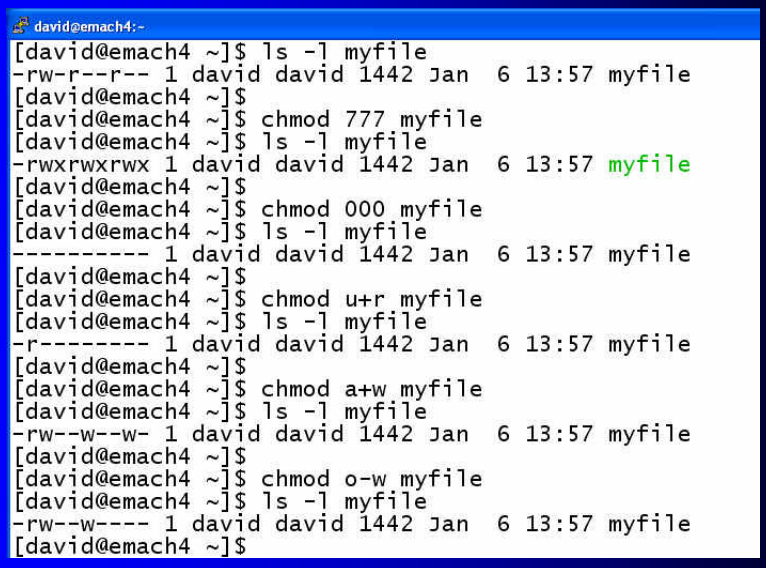
who/**how**/what

- **+** add, other existing permissions unaffected
- **-** remove, other existing permissions unaffected
- **=** set, existing permissions replaced

who/how/**what**

- r - read
- w - write
- x – execute
- s – establish “set id” behavior

chmod – examples



Access decision mechanics

- the actor – which user?
- the file's affiliated user – which is that?
  - if one and the same 1st triplet applies, else
- the file's affiliated group – which is it?
  - if actor in that group 2nd triplet applies, else
- actor is unrelated to file, a "bystander"
  - 3rd triplet applies

Who can read what?

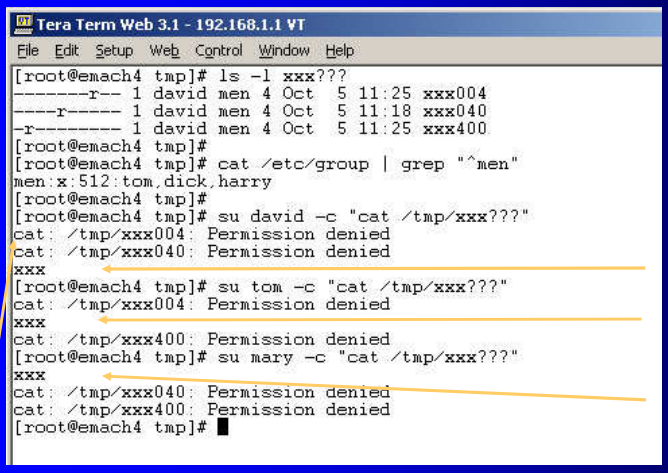
```
[root@EMACH1 schools]# ls -l
total 12
-rw-r--r--   1 root     students     121 Dec  8 17:15 assignments
-rw-rw----   1 root     teachers     119 Dec  8 17:13 grades
-rw-r-----   1 root     administ      95 Dec  8 17:10 salaries
```

socrates (an administrator) can read:
- salaries (because he's an administrator)
- assignments (because bystanders can)

plato (a teacher) can read:
- grades (because he's a teacher)
- assignments (because bystanders can)

aristotle (a student) can read:
- assignments (because he's student)

Permission sets don't overlap



because david is xxx400's affiliated user

because tom is xxx040's affiliated group's member

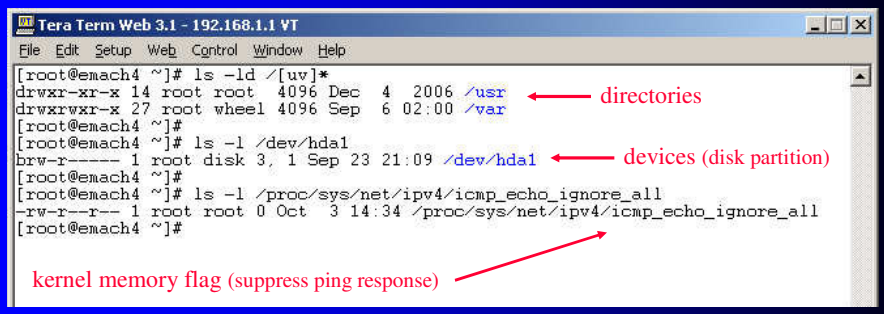
because mary is xxx400's 3rd-party bystander

prohibited! because david is xxx004's affiliated user ("owner")
He is *not* in xxx004's "other" category, which would permit.

Owner more restricted than others, on his own file .

Non-file resources similarly

"everything is a file in unix"



We call it a "file"system but…

"in unix, everything is a file"

			Inode
2	1	2	File type and access rights
2	3	4	Owner identification
4	5	8	File length in bytes
4	9	12	Time of last file access
4	13	16	Time that inode last changed
4	17	20	Time that file contents last changed
4	21	24	Time of file deletion
2	25	26	Group identifier
2	27	28	Hard links counter
4	29	32	Number of data blocks of the file
4	33	36	File flags
4	37	40	Specific operating system information
4	41	44	Pointer to first data block
56	45	100	14 more pointers to data blocks
4	101	104	File version (for NFS)
4	105	108	File access control list
4	109	112	Directory access control list
4	113	116	Fragment address
8	117	128	Specific operating system information

•object types subject
•to management
•(beyond just files)

Bit fields

Constant	Value	Description
-- file format --		
EXT2_S_IFSOCK	0xC000	socket
EXT2_S_IFLNK	0xA000	symbolic link
EXT2_S_IFREG	0x8000	regular file
EXT2_S_IFBLK	0x6000	block device
EXT2_S_IFDIR	0x4000	directory
EXT2_S_IFCHR	0x2000	character device
EXT2_S_IFIFO	0x1000	fifo
-- process execution user/group override --		
EXT2_S_ISUID	0x0800	Set process User ID
EXT2_S_ISGID	0x0400	Set process Group ID
EXT2_S_ISVTX	0x0200	sticky bit
-- access rights --		
EXT2_S_IRUSR	0x0100	user read
EXT2_S_IWUSR	0x0080	user write
EXT2_S_IXUSR	0x0040	user execute
EXT2_S_IRGRP	0x0020	group read
EXT2_S_IWGRP	0x0010	group write
EXT2_S_IXGRP	0x0008	group execute
EXT2_S_IROTH	0x0004	others read
EXT2_S_IWOTH	0x0002	others write
EXT2_S_IXOTH	0x0001	others execute

How to extend permission to…

- a certain group, plus one other guy (who doesn't belong in it) ?
- two groups? three?
- miscellaneous ungrouped users?

Access contol lists (ACLs)

- ACLs extend the rules
  - "to define more fine-grained discretionary access rights" ACL man page
  - apply arbitrary permissions for arbitrary users on arbitrary files in any combination
- ACLs reside in the filesystem (ext2)
  - each file can have its own
- for users in a file's ACL
  - ACL's triplet eclipses/replaces permission string's
- for any others
  - permission string's sub-triplet still governs unaffected

Access contol lists (ACLs)