

# Authorization

February 10, 2012

## Administrative – submittal instructions

- answer the lab assignment's questions. This week the Word document answer file is supplied. Download (from the instructions), fill in, submit.
- email to [csci530l@usc.edu](mailto:csci530l@usc.edu)
- exact subject title must be "authorizationlab"
- deadline is start of your lab session the following week
- reports not accepted (zero for lab) if
  - late
  - you did not attend the lab (except DEN or prior arrangement)
  - email subject title deviates

## Review

- Authentication: proving the identity of someone
- Authorization: allowing a user to access certain resources

## Government authorization


- documents have “classifications”
- employees have “clearances”
  - confidential
  - secret
  - top secret

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

$$\text{access decision} = \text{Ⓢ/Ⓢ} = f(\text{document's classification, clearance})$$

## 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

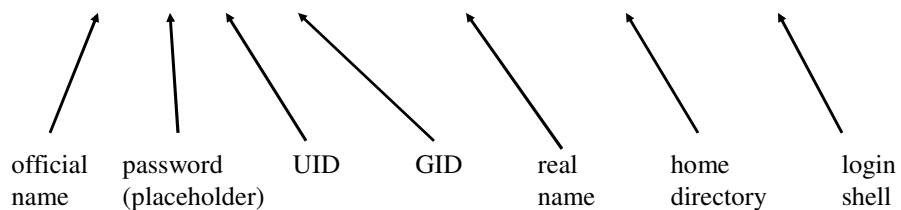
- system keeps a list of user accounts
- system usage demands a user identification
  - supplied at login... no login, no usage
- a user id is implicit in all session activities
  - all session activities are performed by processes
  - every process has some user id as an attribute
  - helps determine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

## The files of record

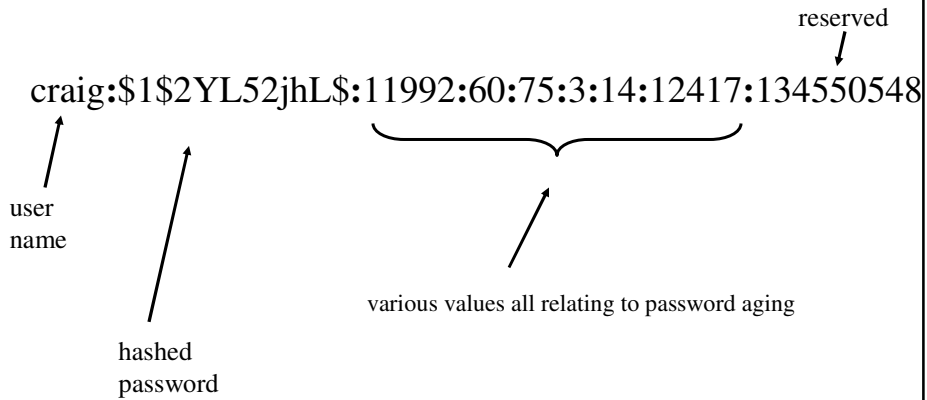
- /etc/passwd – holds list of recognized users
- /etc/shadow – holds their passwords
- /etc/group – holds list of recognized groups, names of member users for each

## /etc/passwd entries hold user information

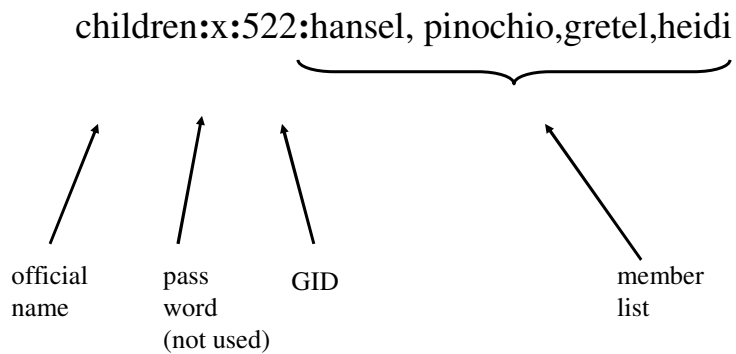
craig:x:507:507:Craig Smith:/home/craig:/bin/bash



## /etc/shadow entries hold ancillary user information



## /etc/group entries hold group information



## Adding and deleting users

- adding
  - “useradd” command
  - then set password with “passwd” command
- deleting
  - “userdel -r” command ( -r removes home directory )

## Adding users in 2 steps

```
[root@EMACH1 /root]# useradd charlie ← step 1
[root@EMACH1 /root]# passwd charlie ← step 2
Changing password for user charlie
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
Now find out what happened!
↓
[root@EMACH1 /root]# su charlie ← become charlie
[charlie@EMACH1 /root]$ cd ← enter his home directory
[charlie@EMACH1 charlie]$ pwd ← identify home directory
/home/charlie
[charlie@EMACH1 charlie]$ ls -a ← directory is populated
. .Xdefaults .bash_profile .kde .screenrc
.. .bash_logout .bashrc .kderc Desktop
[charlie@EMACH1 charlie]$ cat /etc/passwd | grep charlie
charlie:x:531:539:./home/charlie:/bin/bash ← charlie's in the list alright
```

## Deleting users

```
[root@EMACH1 /root]# userdel -r charlie
[root@EMACH1 /root]# su charlie
su: user charlie does not exist ← doesn't live here anymore
[root@EMACH1 /root]# ls -a /home/charlie
ls: /home/charlie: No such file or directory ← home directory who??
[root@EMACH1 /root]# cat /etc/passwd | grep charlie
[root@EMACH1 /root]# ← gone. really!
```

## Groups

- Purpose
  - Let a set of users share files by extending common permissions to them
- Mechanism
  - Files have a group affiliation
  - Users have group memberships
  - Separate access to a file can be extended to members of its group

# There are groups

Groups are defined in /etc/group

file /etc/group

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

# Composing a group

- assign groups to users

- use usermod

```
usermod -G employees,salesmen willie
```

- or, assign users to groups

- use gpasswd

```
gpasswd -a willie employees
```

```
gpasswd -a willie salesmen
```

```
}
```

same  
result

```
gpasswd -M willie,billy,milly fools
```

## Files have (1) a user affiliation

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

```
[root@EMACH1 schools]# ls -l          Files
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
```

Their affiliated users

## Files have (2) a group affiliation

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

```
[root@EMACH1 schools]# ls -l          Files
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
```

Their affiliated groups

## Files have (3) a permissions setting

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

```
[root@EMACH1 schools]# ls -l
```

Permissions	Count	Owner	Group	Size	Date	Time	File Name
-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

total 12

Files

Their permissions settings

## Where? : inode structure of a file found in inode table of an ext2 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	
4	113	116	Fragment address	
8	117	124	Specific operating system information	

## 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

One of 3 permissions triplets applies in any given case


**-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**

## Meaning for files


letter  : -or else-

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


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  : -or else-

- **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

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

```
[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
```

Diagram illustrating the mapping of file permissions to commands:

- `chmod` is associated with the permissions column.
- `chown` is associated with the owner column.
- `chgrp` is associated with the group column.

## 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

<b>- - -</b>	<b>0 0 0</b>	<b>0</b>	Used in triples: e.g., 750 = rwxr-x---
<b>- - x</b>	<b>0 0 1</b>	<b>1</b>	
<b>- w -</b>	<b>0 1 0</b>	<b>2</b>	
<b>- w x</b>	<b>0 1 1</b>	<b>3</b>	
<b>r - -</b>	<b>1 0 0</b>	<b>4</b>	
<b>r - x</b>	<b>1 0 1</b>	<b>5</b>	
<b>r w -</b>	<b>1 1 0</b>	<b>6</b>	
<b>r w x</b>	<b>1 1 1</b>	<b>7</b>	

## changing just some permissions – who/how/what specification

<b>who</b>	<b>how</b>	<b>what</b>
<b>u</b>	<b>+</b>	<b>r</b>
<b>g</b>	<b>-</b>	<b>w</b>
<b>o</b>	<b>=</b>	<b>x</b>
<b>a</b>		<b>s</b>

## 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

```
david@emach4:~$ ls -l myfile
-rw-r--r-- 1 david david 1442 Jan  6 13:57 myfile
[david@emach4 ~]$
[david@emach4 ~]$ chmod 777 myfile
[david@emach4 ~]$ ls -l myfile
-rwxrwxrwx 1 david david 1442 Jan  6 13:57 myfile
[david@emach4 ~]$
[david@emach4 ~]$ chmod 000 myfile
[david@emach4 ~]$ ls -l myfile
----- 1 david david 1442 Jan  6 13:57 myfile
[david@emach4 ~]$
[david@emach4 ~]$ chmod u+r myfile
[david@emach4 ~]$ ls -l myfile
-r----- 1 david david 1442 Jan  6 13:57 myfile
[david@emach4 ~]$
[david@emach4 ~]$ chmod a+w myfile
[david@emach4 ~]$ ls -l myfile
-rw--w--w- 1 david david 1442 Jan  6 13:57 myfile
[david@emach4 ~]$
[david@emach4 ~]$ chmod o-w myfile
[david@emach4 ~]$ ls -l myfile
-rw--w---- 1 david david 1442 Jan  6 13:57 myfile
[david@emach4 ~]$
```

## 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

```
Tera Term Web 3.1 - 192.168.1.1 VT
File Edit Setup Web Control Window Help
[root@emach4 tmp]# ls -l xxx???
-----r-- 1 david men 4 Oct  5 11:25 xxx004
----r---- 1 david men 4 Oct  5 11:18 xxx040
-r----- 1 david men 4 Oct  5 11:25 xxx400
[root@emach4 tmp]#
[root@emach4 tmp]# cat /etc/group | grep "^men"
men:x:512:tom,dick,harry
[root@emach4 tmp]#
[root@emach4 tmp]# su david -c "cat /tmp/xxx???"
cat: /tmp/xxx004: Permission denied
cat: /tmp/xxx040: Permission denied
xxx
[root@emach4 tmp]# su tom -c "cat /tmp/xxx???"
cat: /tmp/xxx004: Permission denied
xxx
cat: /tmp/xxx400: Permission denied
[root@emach4 tmp]# su mary -c "cat /tmp/xxx???"
xxx
cat: /tmp/xxx040: Permission denied
cat: /tmp/xxx400: Permission denied
[root@emach4 tmp]# █
```

because david is xxx400's affiliated user

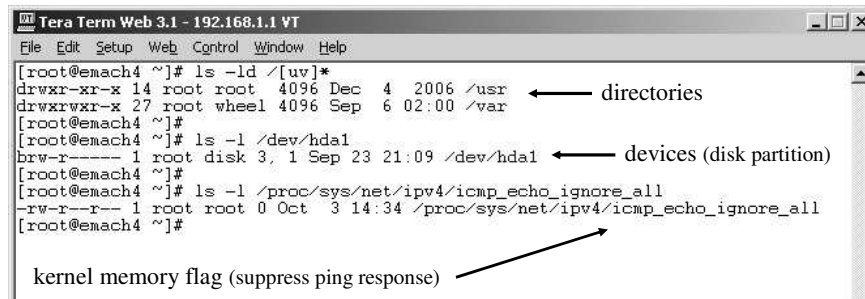
because tom is xxx040's affiliated group's member

because mary is xxx400's 3<sup>rd</sup>-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”



```
Tera Term Web 3.1 - 192.168.1.1 VT
File Edit Setup Web Control Window Help
[root@emach4 ~]# ls -ld /[uv]*
drwxr-xr-x 14 root root 4096 Dec  4 2006 /usr ← directories
drwxrwxr-x 27 root wheel 4096 Sep  6 02:00 /var
[root@emach4 ~]#
[root@emach4 ~]# ls -l /dev/hda1
brw-r----- 1 root disk 3, 1 Sep 23 21:09 /dev/hda1 ← devices (disk partition)
[root@emach4 ~]#
[root@emach4 ~]# ls -l /proc/sys/net/ipv4/icmp_echo_ignore_all
-rw-r--r-- 1 root root 0 Oct  3 14:34 /proc/sys/net/ipv4/icmp_echo_ignore_all
[root@emach4 ~]#
kernel memory flag (suppress ping response) ↗
```

## 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 control 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 control lists (ACLs)

```
root@localhost:~/schools
[root@localhost schools]# ls -l
total 16
-rw-r--r-- 1 root students 171 2008-10-16 06:49 assignments
-rw-rw--- 1 root teachers 44 2008-10-16 06:58 grades
-rw-r----- 1 root administrators 517 2008-10-16 06:49 salaries
[root@localhost schools]#
[root@localhost schools]# tail -3 /etc/group
administrators:x:507:socrates
teachers:x:508:plato
students:x:509:aristotle
[root@localhost schools]#
[root@localhost schools]# su aristotle -c "cat grades"
cat: grades: Permission denied
[root@localhost schools]# su plato -c "cat grades"
assignment1 A; assignment2 C; assignment3 B
[root@localhost schools]#
[root@localhost schools]# setfacl --modify u:aristotle:rw- grades
[root@localhost schools]# setfacl --modify u:plato:--- grades
[root@localhost schools]#
[root@localhost schools]# su aristotle -c "cat grades"
assignment1 A; assignment2 C; assignment3 B
[root@localhost schools]# su plato -c "cat grades"
cat: grades: Permission denied
[root@localhost schools]#
[root@localhost schools]# getfacl grades
# file: grades
# owner: root
# group: teachers
user::rw-
user:plato:---
user:aristotle:rw-
group::rw-
mask::rw-
other::---
grades' ACL
```

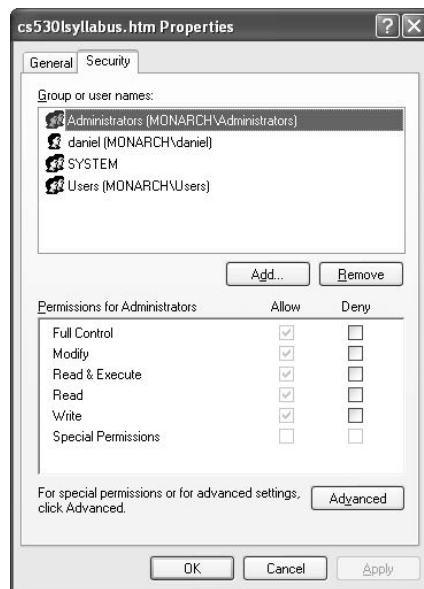
ACL exists for this file

student can't read grades, teacher can

make special changes, via ACL

student can now read grades, teacher no longer can (ACL overrides)

## ntfs filesystem's authorization



## OS, application may impose separate authorization layers

- Windows
  - workgroup or domain controls
- xinetd super server
  - hosts.allow, hosts.deny files
- apache web server
  - per-page or per-directory controls
    - .htaccess files; directives like allow from, deny from
- filesystem controls are independent of these, more fundamental, inviolable

## Notes for this week's lab

- read the instructions
- before you go