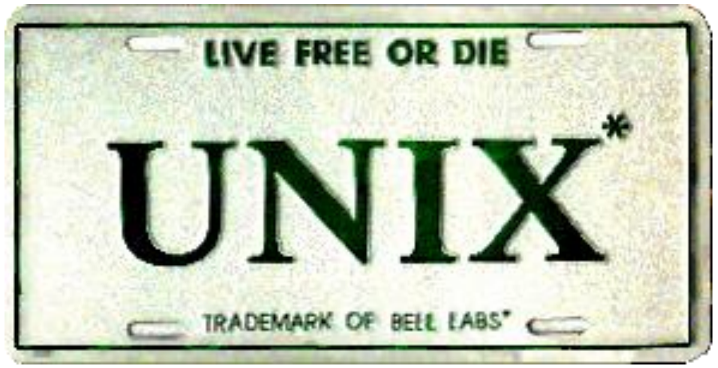

UNIX for Programmers and Users

"UNIX for Programmers and Users"
Third Edition, Prentice-Hall, GRAHAM GLASS, KING ABLES

Slides adapted from Prof. Andrzej (AJ) Bieszczad

File Permissions (Security)

- File permissions are the basis for file security. They are given in three clusters. In the example, the permission settings are "rw-r--r--":

```
-rw-r--r--   1  glass    cs    213   Jan  31  00:12   heart.final
```

User (owner)	Group	Others
rw-	r--	r--

clusters

Each cluster of three letters has the same format:

Read permission	Write permission	Execute permission
r	w	x

File Permission

The meaning of the read, write, and execute permissions depends on the type of file:

	Regular file	Directory file	Special file
Read	read the contents	read the directory (list the names of files that it contains)	read from the file using the read() system call.
Write	change the contents	Add files to the directory	write to the file using the write() system calls.
Execute	execute the file if the file is a program	access files in the directory	No meaning.

File Security

- When a process executes, it has four values related to file permissions:
    1. A real user ID
    2. An effective user ID
    3. A real group ID
    4. An effective group ID

- When you log in, your login shell process has its real and effective user IDs set to your own user ID and its real and effective group IDs set to your group ID.

- When a process runs, the file permissions apply as follows:

- If the process' effective user ID is the same as the owner of the file, the **User** permission apply.

- If the process' effective user ID is different from the owner of the file, but its effective group ID matches the file's group ID, then the Group permissions apply.

File Security

- If neither the process's effective user ID nor the process' effective group ID matches the owner of the file and the file's group ID, respectively, the Others permission apply.
- When an executable with "**set user ID**" permission is executed, the process' effective user ID becomes that of the executables.
- Similarly, when an executable with "**set group ID**" permission is executed, the process' effective group ID becomes that of the executable.
- "Set user ID" and "set group ID" permissions are indicated by an "s" instead of an "x" in the user and group clusters, respectively.
- They may be set using the **chmod** utility.
- The only way to create a new group is to ask the system administrator to add it.
- After a new group is added, any user who wants to be a part of that group must also ask the system administrator.

Listing Group: groups

- groups

- The groups utility allows you to list all of the groups that you're a member of, and it works like this:

groups   userId

- When invoked with no arguments, the group utility displays a list of all of the groups that you are a member of.

- If the name of a user is specified, a list of the groups to which that user belongs are displayed.

- Example:

```
$ groups                              --> list my groups
cs        music
$ _
```

Changing File Group: chgrp

- Changing a File's group : chgrp

chgrp -R groupname fileName

- The chgrp utility allows a user to change the group of files that he/she owns.
- A super-user can change the group of any file.
- All of the files that follow the groupname argument are affected.
- **The -R option** recursively changes the group of the files in a directory.

Changing File Group: example

```
$ ls -lg heart.final
-rw-r--r--   1   glass   cs   213  Jan  31  00:12  heart.final

$ chgrp  music heart.final                    --> change the group.

$ ls -lg heart.final                          --> confirm it changed.
-rw-r--r--    1   glass   music  213  Jan  31  00:12  heart.final
$ _
```

- The chgrp utility is also used to change the group of a directory.

Change File Permissions: chmod

chmod -R   change fileName

- The chmod utility changes the modes (permissions) of the specified files according to the change parameters, which may take the following forms:

clusterSelection+newPermissions (add permissions)

clusterSelection-newPermissions (subtract permissions)

clusterSelection=newPermissions (assign permissions absolutely)

- where clusterSelection is any combination of:
  - u (user/owner)
  - g (group)
  - o (others)
  - a (all)

  and newPermissions is any combination of
  - r (read)
  - w (write)
  - x (execute)
  - s (set user ID/set group ID)

Changing File Permissions

- Note that changing a directory’s permission settings doesn’t change the settings of the files that it contains.
- The **-R** option recursively changes the modes of the files in directories.

Ex: Remove read permission from groups

```
$ ls -lg heart.final              --> to view the settings before the change.
-rw-r-----   1   glass     music  213  Jan  31  00:12  heart.final

$ chmod g-r heart.final

$ ls -lg heart.final
-rw-------   1   glass     music  213  Jan  31  00:12  heart.final
$ _
```

Changing File Permissions: examples

Requirement	Change parameters
Add group write permission	g+w
Remove user read and write permission	u-rw
Add execute permission for user, group, and others.	a+x
Give the group read permission only.	g=r
Add write permission for user, and remove group read permission.	u+w,g-r

Changing File Permission: examples

- Example:

```
$ cd                              --> change to home directory.
$ ls  -ld   .                     --> list attributes of home directory.
drwxr-xr-x   45    glass    4096   Apr    29    14:35
$ chmod   o-rx                    --> update permissions.
$ ls  -ld   .                     --> confirm.
drwxr-x---     45    glass    4096  Apr  29  14:35
$ _
```

Changing File Permissions Using Octal Numbers

- The chmod utility allows to specify the new permission setting of a file as an octal number.
- Each octal digit represents a permission triplet.

For example, for a file to have the permission settings of rwxr-x---

the octal permission setting would be 750, calculated as follows:

	User	Group	Others
setting	rwx	r-x	--
binary	111	101	000
octal	7	5	0

Changing File Permissions Using Octal Numbers

- The octal permission setting would be supplied to chmod as follows:

```
$ chmod  750   .              --> update permissions.

$ ls  -ld         .           --> confirm.
drwxr-x---      45   glass      4096   Apr  29 14:35
$ _
```

Changing File Owner: chown

```
chown  -R  newUserId  fileName
```

- The chown utility allows a super-user to change the ownership of files.
- Some Unix versions allow the owner of the file to reassign ownership to another user.
- All of the files that follow the newUserId argument are affected.
- The -R option recursively changes the owner of the files in directories.

Changing File Owner: chown

- Example: change the ownership of “heart.final” to “tim” and then back to “glass” again:

```
$ ls  -lg  heart.final      --> to view the owner before the change.
-rw-r-----   1  glass   music  213  Jan 31  00:12  heart.final

$ chown   tim  heart.final      --> change the owner to “tim”.

$ ls -lg heart.final       --> to view the owner after the change.
-rw-r-----   1  tim     music  213  Jan 31  00:12  heart.final

$ chown  glass  heart.final    --> change the owner back to
    “glass”.
$ _
```

Change User Groups: newgrp

- When a process creates a file, the group ID of the file is set to the process' effective group ID.

- Ex: when a file is created by a shell, the group ID of the file is set to the effective group ID of the shell.

- The system administrator chooses which one of the groups is used as login shell's effective group ID.

newgrp [-][groupname]

- The newgrp utility with a groupname as an argument, creates a new shell with an effective group ID corresponding to the groupname.

- The old shell sleeps until the termination of the newly created shell.

- User must be a member of the specified group.

- If the argument is a dash(-) instead of a groupname, a shell is created with the same settings as those of the shell that was created by logging into the system.

Changing Groups: example

```
$ date > test1                  --> create from a "cs" group shell.

$ newgrp  music                 --> create a "music" group shell.

$ date > test2           --> create from a "music" group shell.
^D

$ ls  -lg   test1   test2       --> look at each file's attributes.
-rw-r--r--     1   glass   cs      29  Jan  31   22:57   test1
-rw-r--r--     1   glass   music   29  Jan  31   22:57   test2
$ _
```