Lecture 15: File Manipulators

Lecture Overview

- File security and Ownership issues
- Types of access to files / directories
- Determining file access permission
- Changing file access
  - chmod
  - cnown
  - chgrp
- File management Utilities of UNIX
  - cat
  - cp
  - In
  - mv
  - rm
- comparing and contrasting files
  - comm
  - cmp-
  - diff
  - uniq
- Sending and receiving mails

Gomilitary.in

File Security and Ownership Issues

- System level security
- File level security
  - The user
  - User group
  - Others
- Types of access to files and / or directories
  - Read, Write and Execute permissions
- Changing File access permissions

UNIX, being a multi-user system, incorporates different security measures to prevent unauthorized access. The password facility does provide a security measure to login into the system, but once logged in, a user will have access to everything if the login password is the only security measure.

Security issues can be classified broadly into:

1. Security of files and data from unauthorized users

2. Security of each user's programs and data from every other user.

3. Security of the main Operating System

4. Physical security of the machine.

UNIX can provide security for the first three types, but is not capable of maintaining the physical security of the system.

To protect the users' files and data against unauthorized access, **UNIX** classifies the users and

access permissions into three categories: the **user,** members of the **group** and **others.**

In a typical software development environment, a number of persons work on a single project. These software developers need to share each other's files belonging to their project, but persons not involved with the project should not be able to tamper with their data or files. To cater to this need of software developers, the concept of **groups** has been incorporated in **UNIX.**

Types of access to files and *I* or directories

WRITE : modifying and adding to files

READ : examining file contents

EXECUTE : execution of files as programs

Gomilitary.in

Every file in the file system has three sets of. Read, Write and Execute permissions for the user, the group and others.

Owner is the user who has created the file

Group is the group of one or more users who may access the file as a group.

Others are every body else, using the system, which does not fall in either of the above categories

The groups are denoted by u (for user), g (for group) and 0 (for others) in the chmod command.

.Determining File Access Permission

The three classes of file users (owners groups and others) may be combined with three _ . access (read, write and execute) to give nine possible combinations of permissions.

rwx owner

rwx group

rwx other

The presence of a permission is indicated by the appropriate letter. The absence of a permission is indicated by a hyphen (-) in the corresponding place.

The -I option of the Is command prints the permission information of the files along with other information. The permissions are listed in rwx format for the user, the group and others, in that order.

The leftmost character in the, output of Is -I command indicates the type of the file:

'-' is for ordinary file

'd' is for directory file

'c' is for character device file

'b' is for block device file

The next three characters encode the file owner's read, write and execute permissions.

The next three are the permission for the group and the last set is for everyone else.

Example:

Consider the folowing output of an **Is** -Icommand:

"r ;:.,,-- 3 **ajay 2080 Oct 10 09:34 mydoc.doc**

1 **ajay 3578 Oct 9 11:55 ssad.pln**

rr-rw

i i .i i **i r**

B C D E G- H I

the information displayed is as follows

A) total 391 refers to the total number of disk blocks occupied by the files listed.

B) the type of the file

C) its access permissions

D) number of links to the file (see pg.3.14)

E) the owner of the file

F) the group of the owner of the file

G) the size of the file in bytes

H) the date and the time of last modification and

I) the file name.

Gomilitary.in

The permission for mydoc.doc, **-rwxr-xr--** indicates that, it is an ordinary file and the user has the read, write and execute permission, the members of the group have only read and execute permissions, others can read the file but can not write onto it nor execute it.

For ssad.pln, the user, group members and others have the read and write permissions. No one

can execute this file.

The meanings of the three access modes are different for ordinary files and directory files, which can be understood with the help of the following table:

Access Mode	Ordinary File	Directory File
Read	Allows examination of the file contents	Allows listing of the files within the directory
Write	Allows changing contents of the file	Allows creating new files and removing old ones
Execute	Allows executing file as a command	Allows searching a directory

Changing File Accesses

chmod: change file permission

chown: change file ownership

chgrp: change file group

.**chmod**

$chmod [who] opcode mode file

who : user group others all
opcode : add remove assign
mode: read write execute

$chmod absolute_mode file

read	4
write	2
execute	1

The **chmod (change mode)** utility changes the file access permissions for a file.

Each one of these nine permissions for any of the user can be individually granted or denied with this utility.

Gomilitary.in

In order to use this command the user has to be the owner of that file.

To change permission for each category of users, the file owner has to indicate

Which permission is to be changed.

The file for which the permissions are to be changed.

There are two ways by which this command can be given: **Symbolic** mode and the **Absolute** mode.

In the Symbolic mode, the chmod command, has the following command line format:

$ clunod [who] <opcode> <mode> <file>

The file has to be in the current directory

where:

who: identifies which user can access a. file, where the user can be any of these:

u: file owner (user)

g: group

0: all others

a: all (default)

opcodes: Symbol indicates whether to

+:add

-: take away

=: assign

mode: letters are

r: read

w: write

x: execute

Examples:

$ clunod u+x mydoc.doc <Enter>

where

u	is the user of the file
x	is the execute permission
mydoc,doc	is the name of the file whose permission has to be changed.

$ chmod go+w ssad.pln <Enter>

where

g	is the group user for the file
o	is the other user for the file
w	is the write permission for the file
ssad.pln	is the name of the file

The **Absolute** mode is based on **octal numbers,** representing the three kinds of access permissions (octal numbers include digits 0 through 7, both inclusive)

The **octal** values for read, write and execute are **4,** 2 and 1 respectively, where 4 is for read, 2 _ for write and 1 is for execute.

In order to express the ways in which a particular file is to be accessed, simply add the acta values, that correspond to the individual types of permissions.

Example:

Gomilitary.in

No access		0
Execute access only		1
Write access only		2
Read access only		3
Write and execute	2+1 =	4
Read and execute access	4+1=	5
Read and write	4+2 =	6
Read, Write and Execute	4+2+1=	7

The command line format for chmod with the absolute format is:

$ chmod <mode> <file>

The octal value **777** indicates all access modes for all system users.

octal value **600** indicates read and write, peimission for the file owner, and no access permissiocs for group or other user categories

$ chmod 400 mydoc.doc <Enter>

will have read permission for the owner and no permission for the group and others, fer the file mydoc.doc. .

Question: Write the command to change the permission of the file mydoc.doc to:

read and write for the owner

write and execute for the group

execute only for others

Write the commands both in the symbolic and the absolute modes.

chown

The **chown** (change owner) utility reassigns the ownership of a file from one user to another.

$ chown <ownername> <filename> <Enter>

Example:

$ chown joe ssad.pln <Enter>

will make the user **joe** the owner of the file named ssad.pln

chgrp

The chgrp (change group owner) performs the same function for the group that owns the file.

$ chgrpgroupname filename <Enter>

Example:

$ chgrp finance ssad.pln <Enter>

will change the group of the file ssad.pln to finance.

Note: Only the present owner can change owners and groups.

File Management Utilities of UNIX

- cat: display the contents
- cp: copy a file
- In: create a link
- mv: rename a file
- rm: remove a file
- Comparing and Contrasting files
  - diff
  - comp
  - cmp
  - uniq
- sending and receiving mails
  - mail command

Gomilitary.in

The file management commands help us to perform various operations on files. A large number of system utilities are used to process text or data in files.

These file-processing utilities offer users a wide range of functions such as:

- duplicate, rename or delete files
- splitting and combining files
- compare file contents
- searching and modifying file contents
- sorting file contents

- manipulating tabular data
- process status information

.**cat**

cat (for **concatenate),** is mostly used to display the contents of a specific file on the terminal. The command line format for the cat command as:

$ cat <filename> <Enter>

where

filename is the name of the file to be displayed.

The **cat** command can also be used to display the contents of more than one file by giving the filenames separated by spaces.

The cat command without any file name takes each line from the standard input device (key board) and writes it to the standard output device (the terminal). The effect being that each line typed is echoed back, and to end this press **<CTRL+ D>** simultaneously.

Example:

Display the contents of the file poem1

.**$ cat poeml <Enter>**

This is the season
for forgetfulness
the quiet time
the healing time
till in GOD'S wisdom
you love again
$

To display the contents of two files poem1 and poem2.

$ cat poem1 poem2 <Enter>
This is the season
for forgetfulness
the quiet time
the healing time
till in GOD'S wisdom

you love again.
wishes are old
but are often being told
still they are so warm
and sincere,
and will always be there
for you to care
so that its a pleasure to share

the old memories that are always there

$ cat <Enter>

This is to verify cat command	line typed by user
This is to verify cat command	line echoed by cat
Without any argument	line typed in by user
Without any argument	line echoed by cat
<ctrl> D	to end the command

The cat command supports the "wild card" characters.

.cp

Gomilitary.in

- To make duplicate copies of files
- $ cp file1 file2
- $cp file1 file2 file3……..file n directory

The cp command is used to create duplicate copies of ordinary files.

The general command line format is:

$ cp <file1> <file2> <Enter>

where :file1is the source file, file2 is the destinaition file.

Example:
$ cp poem1 poem3 <Enter>

will make a copy of poem1 into poem3.

- If the target file is an ordinary file, and it already exists, the contents of the target is erased and the new contents are written onto the file.
- If the target file is a directory, source file is copied into that directory with the same name.
- The user must have read permission for the source file and write permission for the target file I directory.
- In case more than two arguments are passed to the cp command, then the last one has to be a directory immediately under the current directory or given with a path.

Example:

To copy the files poem1, poem2 and poem3 to the directory test

$ cp poem1 poem2 poem3 test <Enter>

To refer to files, not present in the current directory, the pathname of the source and the target file (or directory) need to be specified.

Example:

To copy the passwd file from the /etc directory to the user's HOME directory (say /usr/mano), under a new name (see fig):

$ cp /etc/passwd /usr/mano/mypass <Enter>
if the working directory is not mypass

$ cp /etc/passwd mypass <Enter>
if th_ current directory is lusr/mano

Note:

1. Most UNIX systems are so set up that a user can copy unrestricted files into the user's home directory from other directories, but no files can be copied into somebody else's home directory.
2. The cp command supports the "wild card" characters.

.In

- To create multiple links to an existing file
- Linking files across directories
- All links to a file access the same physical file
- Modifying contents from any access changes the file
- $ In <old_file> <linked_file>

Gomilitary.in

The In command adds an additional directory entry (link) for an **ordinary file**. It can be accessed from more than one directory or by more than one name. It creates another name for the same file, called a **link or alias name**, whereas the cp command creates another copy of the file.

The name of the linked file, if in the same directory, must be different, but the files may have the same name if they are linked across directories.

The linked files have the same permission and the same inode number

The command line format for the In command is:

$ In <firstname> <secondname> <Enter>

Where

firstname is the name of the ordinary file for which a link is to be established.

secondname is the additional name of the file specified by the firstname.

The number of links of a file can be viewed by Is -I command. With Is -i option, it can be verified that all the linked files have the same inode number.

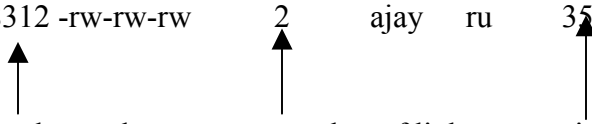
Example:

```
$ In ssad.pln    ssas.lnk        <Enter>        create a link

$ ls -li ssad.*          <Enter>                verify the link and inode number

total 215
8312 -rw-rw-rw       2       ajay    ru      3578    Oct 9 11:55 ssad.pln

8312 -rw-rw-rw       2       ajay    ru      3578    Oct 9 11:55 ssad.lnk
```



inode number number of links size

Note:

1. Only the owner of a file can establish a link.
2. <Filename> may also have a path in it.