Lecture – 15

Getting Started with UNIX

References : Sumitabha Das

SECTION -C

Getting Started with UNIX

References : Sumitabha Das

Introduction

- Changing permission modes
- Absolute & Relative Permissions

Changing file permissions : chmod

- A file or directory is created with a default set of permissions. Generally the default setting write- protects a file from all except user(the owner of the file) though all users may have read access. However this may not be so on your system. To know your system's default settings, create a file xstart:

  $ cat /usr/bin/startx > xstart *actually copies the file startx*

  $ ls – l xstart

  -rw-r- - r - - 1 kumar metal 1906 Sep 5 23:58 xstart

- It seems that, by default a file doesn't also have execute permissions. So how does one execute such file? To do that, the permission of the file need to be changed. This is done with the chmod.
- The **chmod** (change mode) command is used to set the permissions of one or more files for all three categories of users (user, group and others).
- It can run only by the user(the owner) and the super user. The command can be used in two ways:

We'll consider both ways of using chmod in the following sections:

(1) Relative permissions:

when changing permission in a relative manner, chmod only changes the permissions specified in the command line and leaves the other permissions unchanged.

- In this mode it uses the following syntax :

chmod *category operation permission filename(s)*

chmod takes as its arguments an expression comprising some letters and symbols that completely describe the user category and the type of permission being assigned or removed. The expression contains three components :

- ✓ User category (user, group, others)

- By using suitable abbreviations for each of these components, you can frame a compact expression and then use it as an argument to **chmod**. The abbreviations used for these three components are shown in table:
- Abbreviations used by **chmod:**

Category	Operation	Permission
u – User g – Group o – others a – All (ugo)	**+** - Assigns permissions - - Remove permissions = - assign absolute permissions	r -read permission w – write permission x –Execute permission

- Now let's consider an example. To assign execute permission to the user (owner) of the file ***xstart***, we need to frame a suitable expression by using appropriate expression by using appropriate characters from each of the three columns of above table. Since the file need to be executable only by the user, the expression required is **u + x:**

  $ chmod u+x xstart

  $ ls – l xstart

  -rwxr- - r - - 1 kumar metal 1906 May 10:20 : 30 xstart

- The command assigns (+) execute (x) permission to the user (u), but other permissions remain unchanged.
- You can now execute the file if you are the owner of the file but the other categories (i.e. group and others) still can't.
- To enable all of them to execute this file, you have to use multiple characters to represent the user category (ugo):

  $ chmod ugo+x xstart ; ls – l xstart **Or**

  $ chmod a+x xstart *(a implies ugo)* **Or $ chmod +x** *(Bydefault a is implied)*

  -rwxr-xr-x 1 kumar metal 1906 May 10:20 : 30

(2) Absolute Permissions :

- Sometimes you don't need to know what a file's current permissions are , but want to set all nine permission bits explicitly. The expression used by chmod here is a string of three octal numbers(base 8) .
- Octal numbers use the base 8, and octal digits have the values 0 to 7. This means that a set of three bits can represent one octal digit. If we represent the permissions of each category by one octal digit, this is how the permissions can be represented :
  - ✓ Read permission – 4 (Octal 100)
  - ✓ Write permission – 2 (Octal 010)

- For each category we add up the numbers. For instance, 6 represents read and write permissions.

Binary	Octal	Permissions	Significance
000	0	-- -	No permissions
001	1	- - x	Executable only
010	2	- w -	Writable only
011	3	- w x	Writable and executable
100	4	r - -	Readable only
101	5	r - x	Readable and executable
110	6	r w -	Readable and Writable
111	7	r w x	Readable, Writable and executable

- Now you can use a different method :

$ chmod 666 xstart ; ls – l xstart

- rw - rw –rw - 1 kumar metal 1906 May 10 20:30 xstart

The 6 indicates the read and write permissions(4+2)

Filters :Using Both Standard Input and Standard Output

Unix command can be grouped into four categories:

1. Directory-oriented commands like mkdir, rmdir, cd and basic file handling commands like cp, mv and rm use neither standard input nor standard output.
2. Commands like ls, pwd, who etc don't read standard input but they write to standard output..
3. Commands like lp that read standard input but don't write to standard output.(lp command is used for printing a file.)

$ lp xyz.ps

Request id is prl-320 (1 file)

$ _

this command notifies the request id – a combination of a printer name and (prl) and job number (320)

4. Commands like cat, wc, cmp, gzip etc that use both standard input and standard output.

- Command in the fourth category, in UNIX is called FILTERS, and the dual stream handling features makes filters powerful text manipulators.
- NOTE: Most filters can also read directly from files whose names are provided as arguments.
- Let's use **bc** command as a filter this time. Consider this file containing some arithmetic expressions:

```
$ cat calc.txt
2^32            maximum memory on a 32 bit computer.
25 * 50
30* 25 + 15 ^ 2
```

You can redirect bc's standard input to come from this file and save output in yet another :

```
$ bc <calc. txt>  result.txt
$ cat result.txt
  4294967296                    This is 2^32
  1250                           this 25*30
   975                          this is 30*25 + 15 ^ 2
```

Applications in Games

There are lots of fun things and games you can use in UNIX. Most of the ones listed below are local to Brown University. Try each of these commands. Check the man pages or the links below if you have trouble, but note, some of the commands do not have man pages. Have fun!

- banner
- figlet
- WhatsForDinner
- food dilbert
- Forecast
- fortune say
- Xteddy
- xdeady
- BattleTris
- nethack
- Netris
- xbill
- xblast
- xboing
- xroach

Research

- **Unix Commands** CCR's computing resources are primarily Linux based and therefore using them requires a basic understanding of the Unix operating system. Some basic commands are provided below.
- **Basic Unix Commands** CCR Reference Card for Linux/UNIX commands pdf
- Show pathname of current directory: pwd
- List files: ls
- Make a directory: mkdir directory-name
- Change directory: cd directory-name
  - Change directory back to home directory: cd
- Copy a file: cp old-filename new-filename
- View a file:
  - cat filename
  - more filename
  - less filename
- Edit a file:
  - emacs filename
  - vi filename
- Delete a file: rm filename
  - Delete a directory (recursively): rm -R directory-name
    - All files and subdirectories are deleted
- Move a file: mv old-filename new-filename
- Change permissions:
  - Arguments to chmod command: ugo+-rwx
    - where ugo are user, group and other; rwx are read, write and execute
  - Add execute permission for yourself: chmod u+x filename
  - Remove read, write and execute for group and other from a directory its contents:
    chmod -R go-rwx directory-name