UNIX Commands 3

Rakesh Yadav
Associate professor
Department of computer science
Shivaji college, university of Delhi
RAKESHYADAV@SHIVAJI.DU.AC.IN

Unix Directories

- A directory is a file whose sole job is to store file names and related information.
- All files whether ordinary, special, or directory, are contained in directories.
- UNIX uses a hierarchical structure for organizing files and directories.
- This structure is often referred to as a directory tree .
- The tree has a single root node, the slash character (/).
- Home Directory:
- The directory in which you find yourself when you first login is called your home directory.
- You will be doing much of your work in your home directory and subdirectories that you'll be creating to organize your files.
- You can go in your home directory anytime using the following command:
- $cd ~
- Here ~ indicates home directory. If you want to go in any other user's home directory then use the following command:
- $cd ~username
- To go in your last directory you can use following command:
- $cd -

Absolute Pathnames:

- Directories are arranged in a hierarchy with root (/) at the top.
- The position of any file within the hierarchy is described by its pathname.
- Elements of a pathname are separated by a /.
- A pathname is absolute if it is described in relation to root, so absolute pathnames always begin with a /.
- These are some example of absolute filenames.
- /etc/passwd
- /users/sjones/chem/notes
- /dev/rdsk/Os3

Relative Pathnames

- A pathname can also be relative to your current working directory.
- Relative pathnames never begin with /.
- Relative to user amrood' home directory, some pathnames might look like this:
- chem/notes
- personal/res
- To determine where you are within the file system hierarchy at any time, enter the command pwd to print the current working directory:
- $pwd

Listing Directories

- To list the files in a directory you can use the following syntax:
- $ls dirname
- Following is the example to list all the files contained in /usr/local directory:
- $ls /usr/local

Creating Directories:

- Directories are created by the following command:
- $mkdir dirname
- Here, directory is the absolute or relative pathname of the directory you want to create. For example, the command.
- $mkdir mydir
- Creates the directory mydir in the current directory. Here is another example:
- $mkdir /tmp/test-dir
- This command creates the directory test-dir in the /tmp directory. The mkdir command produces no output if it successfully creates the requested directory.
- If you give more than one directory on the command line, mkdir creates each of the directories. For example:
- $mkdir docs pub
- Creates the directories docs and pub under the current directory.

Creating Parent Directories

- Sometimes when you want to create a directory, its parent directory or directories might not exist.
- In this case, mkdir issues an error message as follows:
- $mkdir /tmp/amrood/test
- mkdir: Failed to make directory "/tmp/amrood/test";
- No such file or directory
- In such cases, you can specify the -p option to the mkdir command. It creates all the necessary directories for you. For example:
- $mkdir -p /tmp/amrood/test
- Here -p command creates all the required parent directories

Removing Directories

- Directories can be deleted using the rmdir command as follows
- $rmdir dirname
- Note: To remove a directory make sure it is empty which means there should not be any file or sub-directory inside this directory.
- We can create multiple directories at a time as follows:
- $rmdir dirname1 dirname2 dirname3
- Above command removes the directories dirname1, dirname2, and dirname2 if they are empty.
- The rmdir command produces no output if it is successful.

Changing Directories

- You can use the cd command to do more than change to a home directory: You can use it to change to any directory by specifying a valid absolute or relative path. The syntax is as follows:
- $cd dirname
- Here, dirname is the name of the directory that you want to change to. For example, the command:
- $cd /usr/local/bin
- Changes to the directory /usr/local/bin. From this directory you can cd to the directory /usr/home/amrood using the following relative path:
- $cd ../../home/amrood

Renaming Directories:

- The mv (move) command can also be used to rename a directory. The syntax is as follows:
- $mv olddir newdir
- You can rename a directory mydir to yourdir as follows:
- $mv mydir yourdir

The directories . (dot) and .. (dot dot)

- The filename . (dot) represents the current working directory; and the filename .. (dot dot) represent the directory one level above the current working directory, often referred to as the parent directory.
- If we enter the command to show a listing of the current working directories files and use the -a option to list all the files and the -l option provides the long listing.
- $ls –la

```
tty [OPTION]....
```

TTY Command

- The **tty** command of terminal basically prints the file name of the terminal connected to standard input. **tty** is short of teletype, but popularly known as a terminal it allows you to interact with the system by passing on the data (you input) to the system, and displaying the output produced by the system.
- tty [option]
- tty **-s, –silent, –quiet :** Prints nothing, only returns an exit status.
- tty **–help** : It will display the help message and exit.
- tty **–version :** Prints the version information and exits.

touch command

- The ***touch*** command is a standard command used in UNIX/Linux operating system which is used to create, change and modify timestamps of a file.
- Basically, there are two different commands to create a file in the Linux system which is as follows:
- **cat command**: It is used to create the file with content.
- **touch command:** It is used to create a file without any content. The file created using touch command is empty. This command can be used when the user doesn't have data to store at the time of file creation.
- **Touch command Syntax to create a new file:** You can create a single file at a time using touch command.
- $touch filename
- **To create multiple files:** Touch command can be used to create the multiple numbers of files at the same time. These files would be empty while creation.
- `$touch File1_name File2_name File3_name`

.....

- **touch -a**: This command is used to change access time only. To change or update the last access or modification times of a file touch -a command is used ($**touch –a filename).**
- **touch -c** : This command is used to check whether a file is created or not. If not created then don't create it. This command avoids creating files. ($**touch –c filename).**
- **touch -c-d** : This is used to update access and modification time ($**touch –c -d filename).**
- **touch -m** : This is used to change the modification time only. It only updates last modification time ($**touch –m filename).**
- **touch -r** : This command is used to use the timestamp of another file. Here *Doc2* file is updated with the time stamp of File 1.
- touch -r second_file_name first_file_name
- **touch -t** : This is used to create a file using a specified time.
- `touch -t YYMMDDHHMM fileName`

Cat command

- Cat(concatenate) command is very frequently used in Linux. It reads data from the file and gives their content as output.
- It helps us to create, view, concatenate files. So let us see some frequently used cat commands.
- **1. To view a single file**
- $cat filename
- 2. **To view multiple files**
- $cat file1 file2
- 3. **To view contents of a file preceding with line numbers.**
- $cat –n/b filename

4. **Create a file : $cat > filename**

5. Copy the contents of one file to another file.:

```
$cat [filename-whose-contents-is-to-be-copied] > [destination-filename]
```

6. Cat command can suppress repeated empty lines in output

```
$cat -s test.txt
```

7 Cat command can append the contents of one file to the end of another file.

```
$cat file1 >> file2
```

8. Cat command can display content in reverse order using tac command.

$tac filename

9. Cat command can highlight the end of line.

```
$cat -E "filename"
```

- **10 If you want to use the -v, -E and -T option together, then instead of writing -vET in the command, you can just use the -A command line option.**
- `$cat -A "filename"`
- **11 Cat command to open dashed files.**
- `$cat -- "-dashfile"`
- **12. Cat command if the file has a lot of content and can't fit in the terminal.**
- `$cat "filename" | more`
- **13. Cat command to merge the contents of multiple files.**
  `$cat "filename1" "filename2" "filename3" > "merged_filename"`
- **14. Cat command to display the content of all text files in the folder.**
- `$cat *.txt`

date command

- **Date** command is used to display the system date and time.
- Date command is also used to set date and time of the system.
- By default the date command displays the date in the time zone on which unix/linux operating system is configured.
- You must be the super-user (root) to change the date and time.
- **Syntax:**
- `date [OPTION]... [+FORMAT]`
- `date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]`

date command

- 1: **date (no option) : $date**
- 2:**-u Option:** Displays the time in GMT(Greenwich Mean Time)/UTC(Coordinated Universal Time)time zone. $date –u
- 3: **–date or -d Option:** Displays the given date string in the format of date. But this will not affect the system's actual date and time value. Rather it uses the date and time given in the form of string.
- **$date --date=" string "**
- `$date --date="2/02/2010"`
- `$date --date="Feb 2 2010”`
- 4:**Using –date option for displaying past dates:**
- `$date --date="2 year ago"`
- Date and time of 5 seconds ago.
- Date and time of previous day.
- Date and time of 2 months ago.
- Date and time of 10 days ago.

- 5:**Using –date option for displaying future date:**
- `$date --date="next tue"`
- `$date --date="2 day"`
- `$date --date="tomorrow"`
- `$date --date="1 year"`
- 6:**-s or –set Option:** To set the system date and time -s or –set option is used.
- **`$date --set="date to be set"`**
- `$date --set="Tue Nov 13 15:23:34 PDT 2018"`
- `$date`

- 7:**–file or -f Option:** This is used to display the date string present at each line of file in the date and time format.
- This option is similar to –date option but the only difference is that in –date we can only give one date string but in a file we can give multiple date strings at each line.
- **$date --file=file.txt**
- $cat >> datefile
- Sep 23 2019
- Nov 03 2020
- $date --file=datefile
- **Output:**
- Sun Sep 23 00:00:00 PDT 2019
- Tue Nov 3 00:00:00 PDT 2020

List of Format specifiers used with date command:

$date +%[format-option]

- **%D: Display date as mm/dd/yy.**
- %d: Display the day of the month (01 to 31).
- %a: Displays the abbreviated name for weekdays (Sun to Sat).
- %A: Displays full weekdays (Sunday to Saturday).
- %h: Displays abbreviated month name (Jan to Dec).
- %b: Displays abbreviated month name (Jan to Dec).
- %B: Displays full month name(January to December).
- %m: Displays the month of year (01 to 12).
- %y: Displays last two digits of the year(00 to 99).
- %Y: Display four-digit year. %T: Display the time in 24 hour format as HH:MM:SS.
- %H: Display the hour. %M: Display the minute. %S: Display the seconds.

- **Command:**
- $date "+%D"
- **Output:** 10/11/17
- **Command:** $date "+%D %T"
- **Output:** 10/11/17 16:13:27
- **Command:** $date "+%Y-%m-%d"
- **Output:** 2017-10-11
- **Command:** $date "+%Y/%m/%d"
- **Output:** 2017/10/11
- **Command:** $date "+%A %B %d %T %y"
- **Output:** Thursday October 07:54:29 17

The grep Command

- The grep program searches a file or files for lines that have a certain pattern. The syntax is:
- $grep [option] pattern [files]
- The name "grep" derives from the ed (a UNIX line editor) command g/re/p which means "globally search for a regular expression and print all lines containing it."
- A regular expression is either some plain text (a word, for example) and/or special characters used for pattern matching.
- The simplest use of grep is to look for a pattern consisting of a single word. It can be used in a pipe so that only those lines of the input files containing a given string are sent to the standard output. If you don't give grep a filename to read, it reads its standard input; that's the way all filter programs work:
- $ls -l | grep "Aug"

There are various options which you can use along with grep command

Option	Description
-v	Print all lines that do not match pattern.
-n	Print the matched line and its line number.
-l	Print only the names of files with matching lines (letter "l")
-c	Print only the count of matching lines.
-i	Match either upper- or lowercase.
-w	Match whole word
-o	Print only the matched parts of a matching line, with each such part on a separate output line.

- $ cat grep.txt
- $ grep -i "UNix" grep.txt
- $ grep -c "UNix" grep.txt
- $ grep -l "UNix" grep.txt
- $ grep -n "UNix" grep.txt
- $ grep -v "UNix" grep.txt
- grep -o "UNix" grep.txt
- grep -w "UNix" grep.txt

Relative Permissions

- When changing permissions in a relative manner chmod only changes the permissions specified in the command line and leaves the other permissions unchanged.
- Syntax: Chmod category operation permission filenames(s)
- Chmod command is used to set the permission of one or more files for all three category of user.
- Category : User, Group and others (UGO).
- Operation: To be performed assigned and remove permission.
- Permission: Read, Write and Execute.

chmod

- The '$ chmod' command stands for change mode command.
- As there are many modes in Unix that can be used to manipulate files in the Unix environment.
- Basically there are 3 modes that we can use with the 'chmod' command
  1. +w (stands for write and it changes file permissions to write)
  2. +r (stands for read and it changes file permissions to read)
  3. +x (generally it is used to make a file executable)
- `$ chmod +w file.txt`
- `$ chmod +r file.txt`
- `$ chmod +x file.txt`

Abbreviations used by chmod

Category	Operation	Permission
u-User	+ Assigns permission	r- Read
g-Group	- Removes permission	w- Write
o-Others	= Assigns absolute permission	x- Execute
a- All		

- $ chmod u+x kk.txt
- $ ls –l kk.txt
- The command assign(+) execute 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 category (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 kk.txt ; ls –l kk.txt
- $chmod a+x kk.txt :- a implies ugo
- $chmod +x kk.txt :- By default a is implied
- $chmod u+x file file2 file3 file4……
- $chmod go-r kk.txt; ls –l kk.txt
- $chmod a-x, go+r kk.txt
- $chmod o+wx kk.txt; ls –l kk.txt
- $chmod 761 kk.txt
- $chmod 777 kk.txt or $chmod a+rwx kk.txt

Using chmod with Absolute Permissions

Number	Octal Permission Representation	Reference (rwx)
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

References

- Unix: Concepts and Applications, Sumitabha Das, TMH, 4th Edition, 2009.
- https://www.geeksforgeeks.org.
- https://www.tutorialspoint.com.