Managing Files From the Command Line

Goal

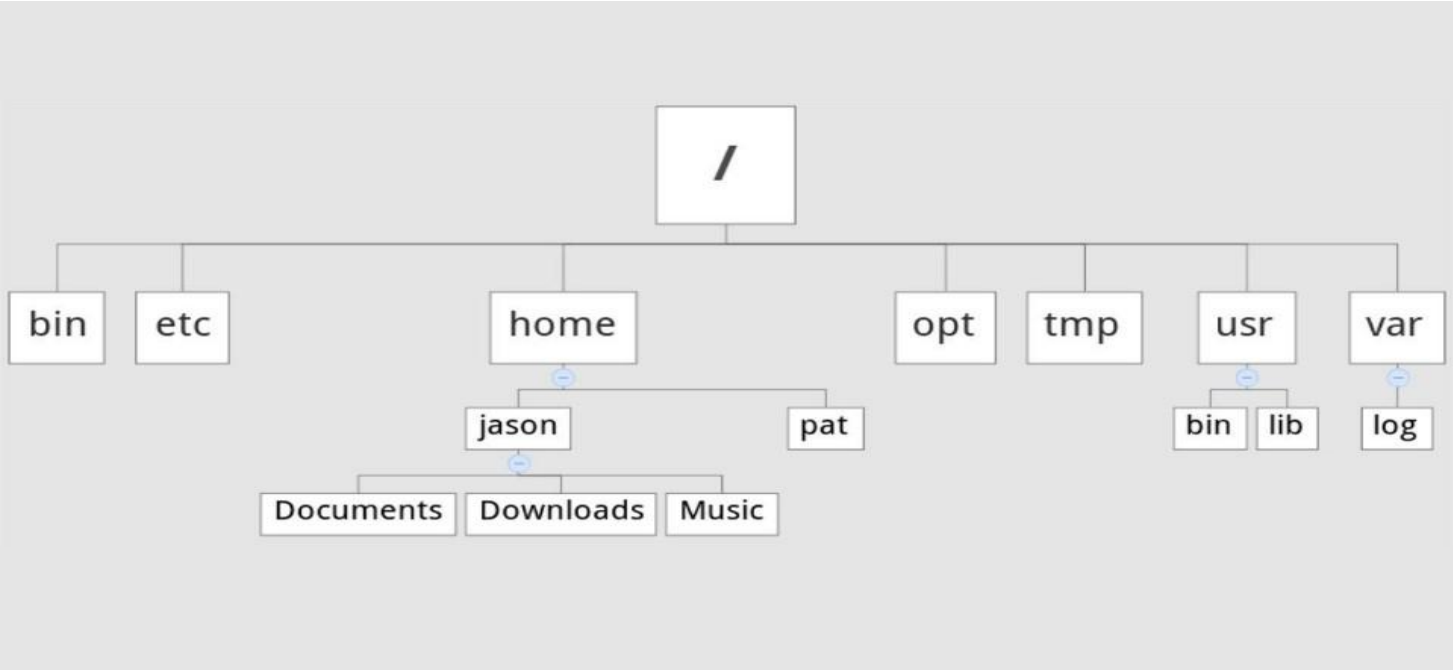
Copy, move, create, delete, and organize files while working from the Bash shell.

Sections

- Describing Linux File-system Hierarchy Concepts.
- Specifying Files by Name.
- Managing Files Using Command-line Tools.
- Making Links Between Files.
- Matching File Names Using Shell Expansions.

Describing Linux File-system Hierarchy Concepts:

All files are stored on file systems, which are organized into a single ***inverted tree*** of directories, known as a file-system hierarchy.



- The **/** directory is the root directory at the top of the file-system hierarchy, its subdirectories are used to group and organize files by type and purpose.

- The most important directories on the system

Directory	Purpose
/usr	Installed software, shared libraries, Important subdirectories include: • **/usr/bin:** User commands. • **/usr/sbin:** System administration commands.
/etc	Configuration files
/var	Variable data specific to this system that should **persist** between boots (databases).
/run	Runtime data for processes started since the **last boot**.
/home	Store user **personal** data.
/root	Home directory for the administrative superuser, **root**.
/tmp	Temporary files. Files which have not been accessed, changed, or modified for 10 days are deleted from this directory automatically.

Specifying Files by Name

Absolute Paths and Relative Paths:

The path of a file or directory specifies its unique file system location. **(**Subdirectories delimited by a forward slash ***/*** **)**

Important

Take care when you deal with space in files/directory names, why?

- **Absolute Paths:**

  Is a fully qualified name, specifying the files exact location in the file system hierarchy, it begins at the root (/) directory listing all subdirectories to the file location "/var/log/messages".

- **The Current Working Directory and Relative Paths**

  The initial location is normally the user's home directory, specifying **only** the path necessary to reach the file from the working directory.

  A path name with anything other than a forward slash as the first character is a relative path name, if user in **"/var"** directory he can use **"log/messages"**.

Navigating Paths:

Command	Purpose
pwd	Displays the full path name of the current working directory
cd	Change your shell's current working directory. If you do not specify any arguments to the command, it will change to your home directory. - option => go to previous directory, **cd .**, **cd ..**
ls	???, -l (long listing format), -a (all files, including hidden files), -R (recursive, to include the contents of all subdirectories).
touch	Normally updates a file's timestamp to the current date and time without otherwise modifying it. Also can create empty files, **how**?
.	Refer to the **current** directory
..	Refer to the **parent** directory

Important

File names beginning with a dot (.) indicate hidden files; you cannot see them in the normal view using **ls** and other commands.

Quiz

1. Which directory contains persistent, system-specific configuration data?
a. /etc
b. /root
c. /run
d. /usr

2. Which directory is the top of the system's file system hierarchy?
a. /etc
b. /
c. /home/root
d. /root

3. Which directory contains user home directories?
a. /
b. /home
c. /root
d. /user

4. Which directory contains temporary files?
a. /tmp
b. /trash
c. /run
d. /var

5. Which directory contains dynamic data, such as for databases and websites?
a. /etc
b. /run
c. /usr
d. /var

6. Which directory is the administrative superuser's home directory?
a. /etc
b. /
c. /home/root
d. /root

7. Which directory contains regular commands and utilities?
a. /commands
b. /run
c. /usr/bin
d. /usr/sbin

8. Which directory contains non-persistent process runtime data?
a. /tmp
b. /etc
c. /run
d. /var

9. Which directory contains installed software programs and libraries?
a. /etc
b. /lib
c. /usr
d. /var

10. Which command is used to return to the current user's home directory, assuming the current working directory is /tmp and their home directory is /home/user?
a. cd
b. cd ..
c. cd .
d. cd *
e. cd /home

11. Which command displays the absolute path name of the current location?
a. cd
b. pwd
c. ls ~
d. ls –d

12. Which command will always return you to the working directory used prior to the current working directory?
a. cd -
b. cd -p
c. cd ~
d. cd ..

13. Which command will always change the working directory up two levels from the current location?
a. cd ~
b. cd ../
c. cd ../..
d. cd -u2

14. Which command lists files in the current location, using a long format, and including hidden files?

a. llong ~

b. ls -a

c. ls -l

d. ls -al

15. Which command will always change the working directory to /bin?

a. cd bin

b. cd /bin

c. cd ~bin

d. cd -bin

16. Which command will always change the working directory to the parent of the current location?

a. cd ~

b. cd ..

c. cd ../..

d. cd -u1

17. Which command will change the working directory to /tmp if the current working directory is /home/student?

a. cd tmp

b. cd ..

c. cd ../../tmp

d. cd ~tmp


BACK IN
5
MINUTES

Managing Files Using Command-line Tools

Common file management commands

Command	Purpose
mkdir directory	Create one or more directories, **-p** option
cp file new-file cp [files] directory	Copy a file Copy files to directory Try with **.** & **..** Take care of overriding files by mistake
cp -r directory new-directory	Copy a directory and its contents
mv file new-file	Move or rename a file or directory How to rename a file ??
rm file	Remove a file
rmdir directory	Remove directory
….	Remove a directory containing files

Making Links Between Files

After completing this section, you should be able to make multiple file names reference the same file using hard links and symbolic (or "soft") links. (remember c++ pointers?)

- **Hard Links and Soft Links**
  It is possible to create multiple names that point to the same file. There are two ways to do this: by creating a hard link to the file, or by creating a soft link (sometimes called a symbolic link) to the file. Each has its advantages and disadvantages.

- **Creating Hard Links**
  Every file starts with a single hard link, from its initial name to the data on the file system. You can create a new hard link (new name) that points to the same data, once created you can't differentiate between the original and the new link.
  You can know number of hard links to a file with ls -l command

Use the **ln** command to create a new hard link (another name) that points to an existing file, commands needs two arguments the 1st is a path to the existing file, the 2nd is a path to the new hard link. **ln newfile.txt /tmp/newfile-hlink2.txt**

- Use **-i** option with **ls** command to display "inode number" of files, files with the same inode numbers and on the same file system are hard links to the same data.

- All hard links that reference the same file will have the same link count, access permissions, user and group ownerships, time stamps, and file content.

- If the original file is deleted the content still available as long as at least one hard link is existing.

- **Limitations of Hard Links**

  Hard links can only be used with regular files. You cannot use **ln** to create a hard link to a directory or special file, also must be on the same file system.
  Use **df** command to list directories on different file systems.

  HW: Why you can't link directories with hard links?

- **Creating Soft Links**

  ln -s command creates a soft link, which is also called a "symbolic link", soft links can link files on different file systems also directories and special files.

  - Create a file
  - Create a soft link using **ln –s**
  - Use ls -l to list both files, what do you notice ??
    1- Number of links is 1 not 2
    2- L instead of – or d for directories

  What will happen when deleting the original file?
  Try **cat** on the soft link.

  What will happen if we created a file with the same name as the original?

• A hard link points a name to data on a storage device
• A soft link points a name to another name that points to data on a storage device

When you make a soft link to a directory then it will act as directory

- Create a soft link for directory
- Navigate to it with **cd** command, what do you notice?
- Try to navigate with **cd -p,** what do you notice?

Matching File Names with Shell Expansions

Objectives

After completing this section, you should be able to efficiently run commands affecting many files by using pattern matching features of the Bash shell.

- **Command-line Expansions**

  Pattern Matching is a shell command-parsing operation that expands a wildcard pattern into a list of matching path names. Command-line **metacharacters** are replaced by the match list prior to command execution.

  Did you hear about 'Regular Expressions'?

Pattern	Matches
*	Any string of zero or more characters.
?	Any single character.
[abc...]	Any one character in the enclosed class (between the square brackets).
[!abc...]	Any one character not in the enclosed class.
[[:alpha:]]	Any alphabetic character.
[[:lower:]]	Any lowercase character.
[[:upper:]]	Any uppercase character.
[[:alnum:]]	Any alphabetic character or digit.
[[:punct:]]	Any printable character not a space or alphanumeric.
[[:digit:]]	Any single digit from 0 to 9.
[[:space:]]	Any single white space character. This may include tabs, newlines, carriage returns, form feeds, or spaces.

Practice to Understand

- **mkdir** testpatterns**; cd** testpatterns
- **touch** alfa bravo charlie delta echo able baker cast dog easy (What this command do?)
- **ls** What is the output?
- **ls a*** What is the output?
- **ls *a*** What is the output?
- **ls ????** What is the output?

- **Tilde Expansion**

  Use the tilde character (~), matches the current user's home directory.

  - **echo ~root** => /root
  - **echo ~student** => / home/student
  - **ls ~/testpatterns** => ????

- **Brace Expansion**

  Brace expansion is used to generate discretionary strings of characters. Braces contain a comma separated list of strings, or a sequence expression. The result includes the text preceding or following the brace definition. Brace expansions may be nested, one inside another. Also double dot syntax **(..)** expands to a sequence such that **{m..p}** will expand to **m n o p**.

  - **echo** Day_{Sunday,Monday,Tuesday,Wednesday}.log What is the output?
  - **echo** StudentID_{1..200}.txt
  - **echo** file{a,b,c}{1,2,3}.txt
  - **echo** file{a{1,2},b,c}.txt

- **Variable Expansion**

  You can assign data as a value to a variable using the following syntax:
  [user@host ~]$ VARIABLENAME=value

  Then you can use the variable's value in your commands

  [user@host ~]$ USERNAME=operator
  [user@host ~]$ echo ${USERNAME}
  operator

- **Command Substitution**

  Command substitution occurs when a command is enclosed in parentheses, and preceded by a dollar sign ($). The $(command) form can nest multiple command expansions inside each other, **echo Today is $(date +%A).**

- **Protecting Arguments from Expansion**

  The backslash **(\)** is an escape character in the Bash shell. It will protect the character immediately following it from expansion.

  [user@host]$ echo The value of $HOME is your home directory.
  [user@host]$ echo The value of \$HOME is your home directory.

  To protect longer character strings, single quotes (') or double quotes (") are used to enclose strings.
  Note that Single quotes stop all shell expansion. Double quotes stop most shell expansion.

  [user@host]$ echo "Will variable $myhost evaluate to $(hostname -s)?"
  [user@host]$ echo 'Will variable $myhost evaluate to $(hostname -s)?'

Quiz

1. Which pattern will match only filenames ending with "b"?

a. b*
b. *b
c. *b*
d. [!b]*

2. Which pattern will match only filenames beginning with "b"?

a. b*
b. *b
c. *b*
d. [!b]*

3. Which pattern will match only filenames where the first character is not "b"?

a. b*
b. *b
c. *b*
d. [!b]*

4. Which pattern will match all filenames containing a "b"?
a. b*
b. *b
c. *b*
d. [!b]*

5. Which pattern will match only filenames that contain a number?
a. *#*
b. *[[:digit:]]*
c. *[digit]*
d. [0-9]

6. Which pattern will match only filenames that begin with an uppercase letter?
a. ^?*
b. ^*
c. [upper]*
d. [[:upper:]]*
e. [[CAP]]*

7. Which pattern will match only filenames at least three characters in length?
a. ???*
b. ???
c. \3*
d. +++*
e. ...*

10 MINUTES

Enjoy the Refreshments

Lab

1- Change your current directory to any directory rather than default one.

2- Create a directory labs/lab1 in your default home directory (don't write full path), if any missing directories let the command create them

3- Create 4 directories named as Season1, Season2, Season3, Season4

4- You have a series of 4 seasons each contains 10 episodes create a folder for each 1 named as season**X**_episode**Y**

5- In one command move all season1 episodes directories to Season1 directory

6- Create terminologies files for a book contains 4 chapters each of them consists of sectionA, sectionB, sectionC, sectionD named as chapter_section

7- List all files of sectionA from all chapters

8- Copy files of chapter1 to new directory called chapter1

9- Remove chapter1

10- Create file testLinks.txt

11- Create a hard links to it called hardLink1.txt, hardLink2.txt

12- View number of links of testLinks.txt and how do you know if files are links to each other

13- Remove testLinks.txt and try to show content of hardLink1.txt