Linux Programming: Assignment-6

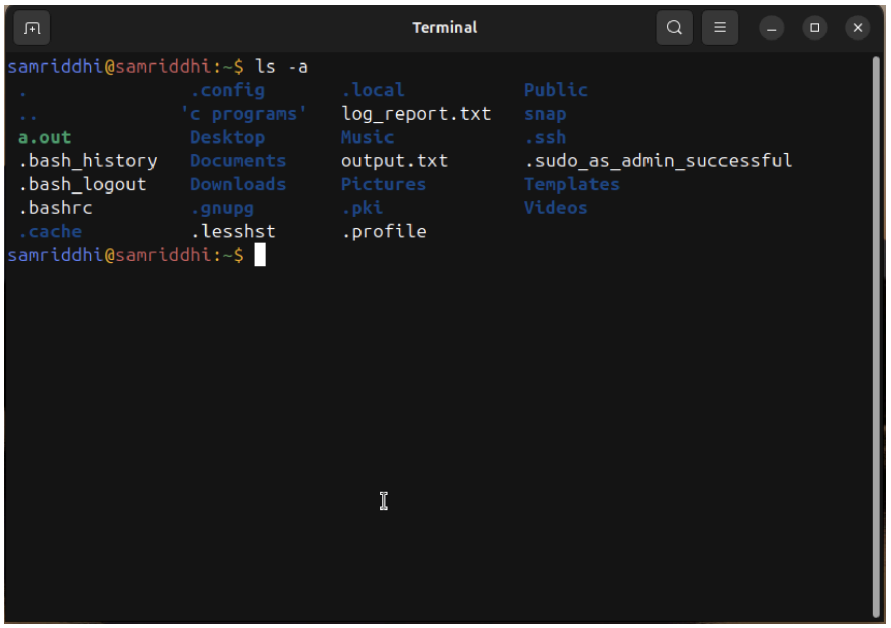
Samriddhi Guha | ENG24CY0157 | 3-C (25)

1. Which command is used to list the contents of a directory? Justify with proper example. (CO1)

The *ls* command is used to display the contents of a directory in Linux. It expands to 'list'.

Syntax: *ls [option]*
Example: *samriddhi@samriddhi:~$ la -a*

The -a option is used to display hidden files in a directory. Such files usually start with (.) symbol.



2. Write the command to create a new directory named 123test_dir. (CO1)

In order to create a new directory, the command *mkdir* is used. It expands to 'make directory'.

To create a new directory named 123test_dir, we can give the following command:

samriddhi@samriddhi:~$mkdir 123test_dir.

This will create a new directory called '123test_dir' in home directory in this case.

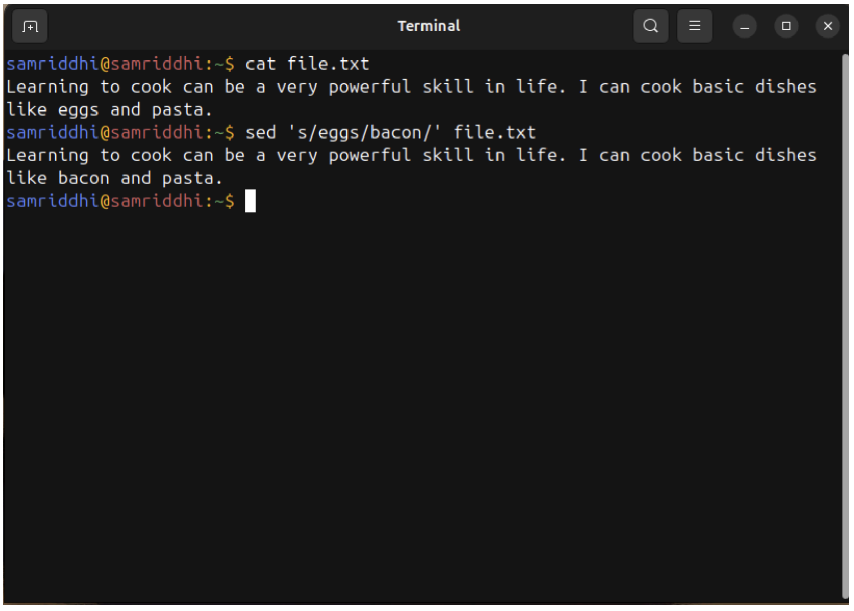
3. What is the purpose of the sed command? Justify with proper example. (CO1)

The SED command in Linux stands for Stream EDitor and is helpful for a myriad of frequently needed operations in text files and streams. Sed helps in operations like selecting the text, substituting text, modifying an original file, adding lines to text, or deleting lines from the text.

Syntax: *sed [options] 'COMMAND' [file_name]*
Example: *samriddhi@samriddhi:~$ sed 's/eggs/bacon' file.txt*

For options,

- -i: Edit files in-place. Without this, sed prints to standard output.
- -n: Suppress automatic printing of pattern space. Used with the p command to print specific lines.
- -e: Add a command to the list of commands to be executed. Useful for multiple commands.
- -f: Read sed commands from a specified file.
- -r or -E: Use extended regular expressions.



4. Which distinct command is used to display one-line descriptions of any commands? (CO1)

The *whatis* command is used to display one-line descriptions of other commands. It searches for the command in a database containing short manual page descriptions.

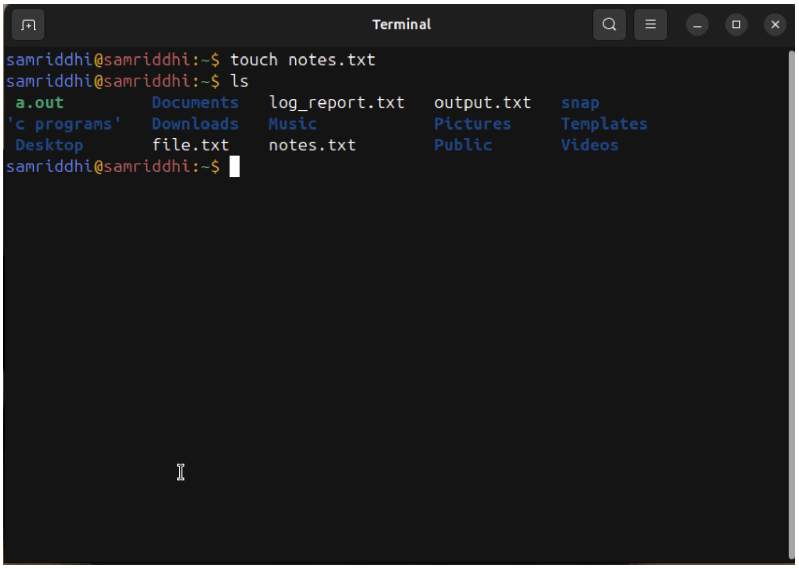
Syntax: *whatis [command_name]*
Example: *samriddhi@samriddhi:~$ whatis mkdir*

5. Write the command to create an empty file named “notes.txt”. (CO1)

To create an empty file named ‘notes.txt’ we use the *touch* command which is used to create new files.

Syntax: *touch [options] [file_name]*

For creating file named ‘notes.txt’,



6. Differentiate between grep and awk commands with an example. (CO2)

grep (Global Regular Expression Print):

- Objective: grep is dedicated to searching text files for a specific pattern and printing the lines that match that pattern. It is a filter for the command line.
- Operation: grep is excellent at quickly finding lines containing a specific string or regular expression. It can provide extra information such as line numbers, number of matches, or just the filenames in which it found a match.
- Ease: grep is usually, also, simpler to use for basic pattern matching.

awk (Aho, Weinberger, Kernighan):

- Purpose: awk is a powerful text processing language made for scanning for patterns and processing them particularly when there is structured data (like a CSV file, or a log file with delimited fields) without further action.
- Functionality: awk can indeed search for patterns, it is primarily a tool to manipulate and transform data matched with a pattern. It can manage the input as a set of records (lines) and fields (columns) allowing for calculations, data extractions, formatting, and reporting. awk is actually a programming language and can use variables, range of conditionals, and loops.
- Complexity: awk is harder to learn and use for simple tasks, but much more flexible while working with more complex data manipulation tasks.

7. Write the command to give read, write, and execute permission to the owner of a file script.sh. (CO1)

To give read, write, and execute permissions to the owner of a file 'script.sh' we can use the *chmod* command.

Syntax: *chmod [u+rwx][file_name]*
Here,

- *u: Represents the user (owner).*
- *+: Adds the specified permissions.*
- *rwx: Represents read, write, and execute permissions.*
- *file_name: The name of the file to modify.*

Example: samriddhi@samriddhi:~$ chmod u+rwx script.sh

8. How is chown different from chgrp? Give one example for each. (CO1)

chown and chgrp both change ownership metadata of filesystem objects, but they operate on different ownership fields and have different typical uses and options.

Core differences:

chown:

- chown (change owner): changes the file's owner (user) and optionally its group.
- Syntax: chown [options] [owner][:][group] [file_name]
- Examples: chown alice file (set owner to alice); chown alice:developers file (set owner alice and group developers).

chgrp:

- chgrp (change group): changes only the file's group.
- Syntax: chgrp [OPTIONS] GROUP file…
- Example: chgrp developers file (set group to developers).

Privileges and behavior:

- Permissions: only root (or a process with CAP_CHOWN) can change the owner to another user. Non-root users can usually change a file's group only if they are a member of the target group and the kernel/distribution permits it.
- chown may also change the group when given; chgrp cannot change the owner.
- Some systems support the "--from" option to change ownership conditionally (available in GNU coreutils).

9. A user complains that they cannot execute a file even though it exists in their directory. How would you troubleshoot this using ls -l, chmod, and whoami? (CO3)

In order to troubleshoot this issue, we will follow these steps:

1. Determine the owner using the *whoami* command.
    - Purpose: Determine the effective username for the user to see what permission category (Owner, Group, or Others) they are in.
2. Permission Check:
    samriddhi@samriddhi:~$ ls -l
3. Command (on the command line): ls -l filename
    - Purpose: To display the current permissions (i.e. -rwxr-x---) associated with the file and who the Owner and Group owner is.
4. Troubleshoot: Determine that the permission set for the user (from step one) has the execute (x) bit set, or not.
5. Add Permission:
    samriddhi@samriddhi:~$ chmod
6. Command (on the command line): use chmod to add the missing execute permission (+x) to the appropriate category (i.e. u, g or o).

    samriddhi@samriddhi:~$chmod u+x filename

    - Purpose: After the user has run the chmod statement, the permission has been altered to allow the user to execute the file, there should no longer be an access error.

10. Design a command pipeline to: find all .log files modified in the last 2 days in /var/log, display them on screen, and save the results into a file recent_logs.txt using tee command. (CO4)

In order to find all .log files modified in the last 2 days in /var/log, we will use the following command:

samriddhi@samriddhi:~$ find /var/log -name ".log" -mtime -2 | tee recent_logs.txt*

The breakdown of this command is as follows:

- find /var/log: The command find is used to find files in a directory hierarchy. This part of the command specifies that the search should be done starting from the directory /var/log.

- -name "*.log": This is a test for the find command that restricts the search to files whose names end with .log.
- -mtime -2: This is a test that restricts the search to files whose last modification occurred less than two days ago. The "- " means "less than" and 2 means periods of 24 hour times
- |: The pipe operator | redirects the standard output from the find command to standard input for the command tee.
- tee recent_logs.txt: The command tee reads from its standard input and writes it to standard output (to the screen) and writes it to the file specified (recent_logs.txt).