Software Lab- 1 (CSL 201) BTech - CSE

Sessional – 1 syllabus (Aug 2019)

Syllabus:

First of all, enable your bash command history. To record all the commands you type in terminal.

→ **To test whether history is enable:**

$ set -o | grep history

if output is "history off"

Then add this line "set – o history**" at end of file** ~/.bashrc

→ Next test these:

echo $HISTFILE

echo $HISTSIZE

echo $HISTFILESIZE

If the first one is blank or /dev/null **Then add this line** "HISTFILE=$HOME/.bash_history" to the end of file ~/.bashrc

If either of last two prints 0 zero, then set them as

$ HISTFILESIZE=500
$ HISTSIZE=500

→ After saving .bashrc, run this

$ source .bashrc

// to activate all changes from current session

1. Linux commands

1. **ls** – ls command is used to list contents of a directory
    1. ls – displays the file in current directory

2. ls –l – displays the file in long formats with its details (size, modified date and time, file and folder owner name etc.)
3. ls –a – list all the hidden files starting with '.'
4. ls -lh – shows the file size in human readable formats
5. ls –F – will add the backslash (/) at the end of each directory
6. ls –r – list the files in the reverse order
7. ls –ltr – displays latest modified file or directory date as last
8. ls –lS – displays the sorted file with big size as first
9. ls --version – shows version of the ls command
10. ls –l /directory name – shows the list of files and folders of the specified directory

2. **cp** - is used for copying files and directories from one location to another.
    1. cp source destination – copies the source file to the target directory

        E.g. cp /home/downloads /home/documents
    2. cp source_file_1 source-file_2 source_file_3 destination_directory – copies the given multiple files at the same time in the same directory
    3. cp –v source_directory destination_directory – shows the verbose output of the files
    4. cp –i source_file destination_folder – copies the files interactively

        i.e. as the user whether to overwrite if already exists

3. **mv** –used to rename the files or directories
    1. mv source_filename target_filename – renames file names
    2. mv source_directory target_directory – renames directory names
4. **rm** – used to remove or delete the file or the directory
    1. rm filename – deletes the given file
    2. rm file1 file2 – deletes more than one file at a time
    3. rm –i filename – interactively deletes the file (asks the user for confirmation)
    4. rm –f filename – forces the deletion of the file if the file is write protected

5. rm –r directory – performs the recursively tree-walk and deletes everything that it finds

6. rm * - deletes the parent directory (if doesn't contains another directory)

7. rm –r * - deletes each and every file and directory present in the parent directory

5. **rmdir** – helps to remove/delete the empty directories

    1. rmdir --version – displays version of the command

    2. rmdir –v dirname – deletes the directory if empty and shows the message

    3. rmdir –p first/second – first remove the child directory and then removes the parent directory

6. **cat** - used to view contents of a file or concatenate files, or data provided on standard input, and display it on the standard output.

    1. cat > filename – creates a new file and waits for the user to the text in the file. ctrl+d or ctrl+z exits from the standard input and writes the contents in the file

    2. cat filename – displays the file contents on the standard output

    3. cat filename.txt | more or | less – used if the contents are more and terminal scrolls very fast. it shows the specified data

    4. cat –n filename.txt – shows each line of the file with the line number

    5. cat filename1; cat filename2; cat filename3 – displays more than one file at a time

    6. cat filename1, filename2 > filename3 – writes the filename1 and filename2 data into the filename3 file (overwrites the contents of filename3 file)

6. a) Touch. - used to create a file. (explore more yourself)

7. **find** - used to find files and directories and perform subsequent operations on them. It supports searching by file, folder, name, creation date, modification date, owner and permissions.

    Syntax - find [where to start searching from] [expression determines what to find] [-options] [what to find]

    1. find ./GFG -name sample.txt - will search the sample.txt in the GFG directory

2. find ./GFG -name *.txt – will search all the files with the .txt extension in the GFG directory
3. find ./GFG -name sample.txt -exec rm -i {} \; - will search the file in the GFG directory when –exec command runs and returns) for successful existence and removes the file using rm command and ask the user for confirmation using the interactive mode
4. find ./GFG –empty – will find all the empty folder and files in the entered directory
5. find . -type f \(-name "*.sh" -o -name "*.txt" -o -name "*.c" \) –
6. . – current directory
7. -type – file type to be found regular files are denoted by f
8. -o – OR operator for multiple matches

8. **chmod** - used to change the access mode of a file.

a) Syntax 1: - chmod [reference][operator][mode] file...

1. reference – are used to distinguish the users to whom the permissions apply
2. operator – specify how the modes of a file should be adjusted
3. mode - indicates which permissions are to be granted or removes from specified classes

Categories:	Reference	Class	Description
	u	owner	file’s owner
	g	group	users who are members of the file’s group
	o	others	users who are neither the file’s owner not members of the file’s group
	a	all	all three of the above
Permissions:		r	permission to read the file
		w	permission to write (or delete) the file
		x	permission to execute the file, or, in the case of a directory, search it.
Operators:		+	Adds the specified modes to the specified classes
		-	Removes the specified modes from the specified classes
		=	The mode specified are to be

		made the exact modes for the specified classes

b) Syntax 2: - chmod [numeric_value] file...

Numeric value:

4	Read
2	Write
1	Execute

E.g. chmod 766 filename – indicates that first number for owner having all rights
(4+2+1 = 7), second number for group (4+2 = 6) and third number for guest (4+2 = 6) and then filename.

9. **wc**- The wc (word count) command in Unix/Linux operating systems is used to find out number of newline count, word count, byte and characters count in a files specified by the file arguments. The syntax of wc command as shown below.

   Syntax: wc [options] filenames

   1. wc -l : Prints the number of lines in a file.
   2. wc -w : prints the number of words in a file.
   3. wc -c : Displays the count of bytes in a file.
   4. wc -m : prints the count of characters from a file.
   5. wc -L : prints only the length of the longest line in a file.

10. **df** – 'df' command stands for "disk filesystem", it is used to get a full summary of available and used disk space usage of the file system on Linux system.

    1. df –a - The same as above, but it also displays information of dummy file systems along with all the file system disk usage and their memory utilization.
    2. df –h - prints the results in human readable format (e.g., 1K 2M 3G).
    3. df – k – displays the file size in kilobytes
    4. df –m – displays the file size in megabytes

11. **du** - used to check the information of disk usage of files and directories on a machine.

1. du /folder/filename - To find out the disk usage summary of a /folder/filename directory tree and each of its sub directories.
2. du – h - provides results in Human Readable Format.
3. du –sh - To get the summary of a grand total disk usage size of a directory use the option “-s” as follows.
4. du – a - displays the disk usage of all the files and directories.
5. du – time - Display the disk usage based on modification of time

12. **netstat**- is a command line tool for monitoring network connections both incoming and outgoing as well as viewing routing tables, interface statistics etc.

    1. netstat - Listing all the LISTENING Ports of TCP and UDP connections
    2. netstat –a | more - Listing all ports (both TCP and UDP) using netstat -a option.
    3. netstat –at - Listing only TCP (Transmission Control Protocol) port connections using netstat -at.
    4. netstat –l - Listing all LISTENING Connections. Listing all active listening ports connections with netstat -l.

13. **ps** - It displays information about a selection of the active processes on the system

    1. top - top is a powerful tool that offers you a dynamic real-time view of a running system
    2. kill - to terminate a process.
    3. killall – if any process have many child processes, this command is used to terminate such process. This command takes the process name as the argument instead of the process id.

14. **more** - used to display the contents of a file in a console. The basic usage of more command is to run the command against a file.

15. **set** - The set command in Linux activates or clears specific flags or settings within the Bash shell environment.

    1. set –c - configures Bash to not overwrite an existing file when output redirection using >, >&, and <> is redirected to a file. By default, Bash allows redirected output to overwrite existing files.

2. –a - Sets all subsequently defined variables for export.
3. –b - Notifies you when background jobs finish running.
4. –e - Tells a noninteractive shell to execute the ERR trap and then exit. This flag is disabled when reading profiles.
5. –f - Disables path name generation.
6. –h - Makes all commands use tracked aliases. (For an explanation of tracked aliases, see the Command execution section in sh.
7. –i - Makes the shell interactive.

16. **Env** - If you enter env with no arguments, it displays the environment variable that it received from its parent (presumably the shell). Arguments of the form variable=value let you add new environment variables or change the value of existing environment variables.

    Syntax - env [-i] [variable=value ...] [command argument ...]

    env [-] [variable=value ...] [command argument ...]

17. **chgrp** - chgrp command is used to change the group ownership of a file. Provide the new group name as its first argument and the name of file as the second argument like this:

    Syntax:
    1. chgrp [OPTION]... GROUP FILE...
    2. chgrp [OPTION]... –reference=RFILE FILE...

    Note: First we need to have administrator permission to add or delete groups. We can Login as root for this purpose or using sudo. In order to add a new group we can use:

    e.g. sudo addgroup group_name

    Options:

    1-c or –changes : To describe the action for each File whose group actually changes.
    2. -f : To suppress error messages.
    3. -v : To describe the action or non-action taken for every File.
    4. –dereference/ –no-dereference: To change the group name of link files.

18. **man** - man command is used to view the on-line reference manual pages for commands/programs like so.

    1. man du
    2. man df

19. **vi**- The default editor that comes with the UNIX operating system is called vi (visual editor). Using vi editor, we can edit an existing file or create a new file from scratch. we can also use this editor to just read a text file.
    1. Syntax: vi filename
    2. vi -R filename : Opens an existing file in read only mode.
    3. view filename : Opens an existing file in read only mode.
    4. Moving within a file:
        1. k : Moves the cursor up one line.
        2. j : Moves the cursor down one line.
        3. h : Moves the cursor to the left one character position.
        4. l : Moves the cursor to the right one character position.
        5. 0 or | : Positions cursor at beginning of line.
        6. $: Positions cursor at end of line.
        7. W : Positions cursor to the next word.
        8. B : Positions cursor to previous word.
        9. (: Positions cursor to beginning of current sentence.
        10.) : Positions cursor to beginning of next sentence.
        11. H : Move to top of screen.
        12. nH : Moves to nth line from the top of the screen.
        13. M : Move to middle of screen.
        14. L : Move to bottom of screen.
        15. nL : Moves to nth line from the bottom of the screen.
        16. colon along with x : Colon followed by a number would position the cursor on line number represented by x.
    5. Control commands (Scrolling)
        1. CTRL+d : Move forward 1/2 screen.
        2. CTRL+f : Move forward one full screen.
        3. CTRL+u : Move backward 1/2 screen.
        4. CTRL+b : Move backward one full screen.
        5. CTRL+e : Moves screen up one line.

6. CTRL+y : Moves screen down one line.
7. CTRL+u : Moves screen up 1/2 page.
8. CTRL+d : Moves screen down 1/2 page.
9. CTRL+b : Moves screen up one page.
10. CTRL+f : Moves screen down one page.
11. CTRL+I : Redraws screen.

6. Editing and inserting in Files(Entering and Replacing Text):
    1. I : Inserts text before current cursor location.
    2. I : Inserts text at beginning of current line.
    3. A : Inserts text after current cursor location.
    4. A : Inserts text at end of current line.
    5. O : Creates a new line for text entry below cursor location.
    6. O : Creates a new line for text entry above cursor location.
    7. R : Replace single character under the cursor with the next character typed.
    8. R : Replaces text from the cursor to right.
    9. S : Replaces single character under the cursor with any number of characters.
    10. S :Replaces entire line.
7. Deleting Characters:
    1. X : Deletes the character under the cursor location.
    2. X : Deletes the character before the cursor location.
    3. Dw : Deletes from the current cursor location to the next word.
    4. d^ : Deletes from current cursor position to the beginning of the line.
    5. d$: Deletes from current cursor position to the end of the line.
    6. Dd : Deletes the line the cursor is on.
8. Copy and Past Commands: Yy : Copies the current line.
    1. 9yy : Yank current line and 9 lines below.
    2. p : Puts the copied text after the cursor.

3. P : Puts the yanked text before the cursor.

9. Save and Exit Commands of the ex Mode :
    1. q : Quit
    2. q! : Quit without saving changes i.e. discard changes.
    3. r fileName : Read data from file called fileName.
    4. wq : Write and quit (save and exit).
    5. w fileName : Write to file called fileName (save as).
    6. w! fileName : Overwrite to file called fileName (save as forcefully).
    7. !cmd : Runs shell commands and returns to Command mode.

20. **tar and remaining untar**- The Linux “tar” stands for tape archive, which is used by large number of Linux/Unix system administrators to deal with tape drives backup. The tar command used to rip a collection of files and directories into highly compressed archive file commonly called tarball or tar, gzip and bzip in Linux. The tar is most widely used command to create compressed archive files and that can be moved easily from one disk to another disk or machine to machine.

    1. Create tar Archive File
    The below example command will create a tar archive file tecmint-14-09-12.tar for a directory /home/tecmint in current working directory. See the example command in action.

    ```
    # tar -cvf tecmint-14-09-12.tar /home/tecmint/
    ```

    2. Create tar.gz Archive File
    To create a compressed gzip archive file we use the option as z. For example the below command will create a compressed MyImages-14-09-12.tar.gz file for the directory /home/MyImages. (Note : tar.gz and tgz both are similar).

    ```
    # tar cvzf MyImages-14-09-12.tar.gz /home/MyImages
    OR
    # tar cvzf MyImages-14-09-12.tgz /home/MyImages
    ```

    3. Untar tar Archive File
    untar or extract a tar file, just issue following command using option x (extract). For example the below command will untar the file public_html-14-09-12.tar in present working directory. If you want to untar in a different directory then use option as -C (specified directory).

    ```
    # Untar files in Current Directory ##
    # tar -xvf public_html-14-09-12.tar
    ```

Untar files in specified Directory
tar -xvf public_html-14-09-12.tar -C /home/public_html/videos/

21. **uuencode**- uuencode(1) - Linux man page. Name. uuencode, uudecode - encode a binary file, or decode its representation. Synopsis. Description. Uuencode and uudecode are used to transmit binary files over transmission mediums that do not support other than simple ASCII data.

22. **find**- The Linux Find Command is one of the most important and frequently used command command-line utility in Unix-like operating systems. Find command is used to search and locate the list of files and directories based on conditions you specify for files that match the arguments. Find can be used in a variety of conditions like you can find files by permissions, users, groups, file type, date, size, and other possible criteria.

    1. Find Files Using Name in Current Directory
       Find all the files whose name is filename.txt in a current working directory.

       # find . -name filename.txt

       ./filename.txt

    2. Find Files Under Home Directory
       Find all the files under /home directory with name tecmint.txt.

       # find /home -name tecmint.txt

       /home/tecmint.txt

    3. Find Directories Using Name
       Find all directories whose name is Tecmint in / directory.

       # find / -type d -name Tecmint

       /Tecmint

    4. History- List Last/All Executed Commands in Linux

23. **Ping**- ping command is used to determine connectivity between hosts on a network (or the Internet):

    Syntax: $ ping google.com

24. **Ifconfig**- fconfig in short “interface configuration” utility for system/network administration in Unix/Linux operating systems to configure, manage and query network interface parameters via command line interface or in a system configuration scripts. The “ifconfig” command is used for displaying current network configuration information, setting up an ip address, netmask or broadcast address to an network interface, creating an alias for network interface, setting up hardware address and enable or disable network interfaces.

    1. ifconfig –a - will display information of all active or inactive network interfaces on server. It displays the results for eth0, lo, sit0 and tun0.

25. **traceroute**- The traceroute command is used in Linux to map the journey that a packet of information undertakes from its source to its destination. One use for traceroute is to locate when data loss occurs throughout a network, which could signify a node that's down. The only parameter you must include when you execute the traceroute command is the host name or IP address of the destination.

    Syntax: $ traceroute

26. **diff**- Normally, to compare two files in Linux, we use the diff – a simple and original Unix command-line tool that shows you the difference between two computer files; compares files line by line and it is easy to use, comes with pre-installed on most if not all Linux distributions.

    1. The conventional syntax for running diff is as follows:

       $ diff [OPTION]… FILES

       $ diff options dir1 dir2

    2. By default, its output is ordered alphabetically by file/subdirectory name as shown in the screenshot below. In this command, the -q switch tells diff to report only when files differ.

       $ diff -q directory-1/ directory-2/

GDB - GDB stands for GNU Project Debugger and is a powerful debugging tool for C. A portable debugger that runs on many Unix-like systems and works for many programming languages. It helps you to poke around inside your C programs while they are executing and also allows you to see what exactly happens when your program crashes. GDB operates on executable files which are binary files produced by compilation process.

Here are few useful commands to get started with gdb for the above example:-

run or r –> executes the program from start to end.
break or b <line no>–> sets breakpoint on a particular line.
disable -> disable a breakpoint.
enable –> enable a disabled breakpoint.
next or n -> executes next line of code, but don't dive into functions.
step –> go to next instruction, diving into the function.
list or l –> displays the code.
print or p –> used to display the stored value.
quit or q –> exits out of gdb.
clear –> to clear all breakpoints.
continue or c –> continue normal execution
set history save on → to save the command history in the specified or default location
set history filename <file name> → to specify the command history file name
core <file name> → to open the saved core dump file with name <file name>
bt → to back track the execution flow from current instance
watch <variable name, can be with condition>→ keep watch the values of that variable, whether that condition met.

Enabling system to store core dump files (by default it is disable):
$ ulimit -c unlimited

Enabling system to store core dump file with specific file name in current folder:
$ sudo bash -c 'echo core.%e.%p.%s > /proc/sys/kernel/core_pattern'
$ cat /proc/sys/kernel/core_pattern

Loading core dump file in gdb mode:
gdb> core <core dump file name>
gdb> bt full

Note:

Ist Evaluation of Lab will be done by making all students to login into the server and execute the sequence of commands – which will be asked with description at the time of exam. And will store all the commands at server under your login. Will evaluate it.