Lab 1 : Basic commands

Sl No.	Question	Command
1.	Display both date and time	**date**
2.	Clearing the screen	**clear**
3.	To display the calendar	**cal**
4.	To display who are the users currently working	**who**
5.	To view processes	**ps**
6.	To list files	**ls, ls -l**
7.	Directing output to a file	**ls > a, cat a**
8.	Counting number of lines in a file	**wc a**
9.	Feeding output of one command to another	**ls | wc, ls | more**
10.	Assign a value to a variable and display the value of that variable	**$x=2 (for int) echo $x $y=”Welcome” (for string) echo $x**
11.	Display a string or a sentence given	**echo “Welcome to MSRIT”**
12.	Locate a command	**type who (displays /usr/bin/who path means external command) type echo (displays built-in means internal command)**
13.	Display the path	**echo $PATH**
14.	What will be displayed if the command is not available?	**Eg. $whw->Command not found**
15.	Find out whether a command is internal or external	**Internal (Built-in): echo,cd,pwd,help,exit External : who.date,ls,cal,cp.mv**
16.	use option for any command	**ls –l, cp –i**
17.	Use any command with more than one argument	**ls –l a, (lists file a) ls –l a b (lists file a and b)**
18.	How can you combine commands and use?	**Echo “These are the files”;ls**
19.	Can a command line overflow to multiple lines. Try example	**$echo “HOW >ARE >U?” prints HOW ARE U? Ans : Yes**
20.	Browse for the manual pages online with navigation and search and -k , apropos and whatis	**man, man –k net, apropos net what is man**
21.	How to handle in Linux when things go wrong	**^u, ^z, ^d**

$script –f <filename> open a spool file to store all command execution
$exit to end the script

Lab 2:General purpose utilities

Sl No.	Action	Command	Output
1.	Use calendar with various options	cal -3 cal 1 1972 cal 2008	**(prev month, cur month, next month calendar Specified month, year calendar Specified year calendar**
2.	display date and time with various options	• $date +"Hi! Date is %D" • $date +"Hi Date is %d/%m/%y" • $date +"Hi Date is %D %n Time is %T	**Prints Hi! Date is mm/dd/yy format Prints Hi Date is dd/mm/yy format %n for new line Prints Date and time in separate lines**
3.	Display a message with escape sequences	$echo "I have $100" $echo –e " I have \$100" $echo –e " Hello \a MSRIT" $echo –e "Hello \n MSRIT"	**Prints I have 00 Prints I have $100 Print Hello <sound> MSRIT Prints Hello MSRIT**
4.	Use the alternative to echo	$printf "HELLO" $printf "Hello Everybody\n" $y=MSRIT $printf "Welcome to %s\n" $y	**Print HELLO Prints Hello Everybody new line Prints Welcome to MSRIT**

5.	Use calculator	bc	**bc
			ibase= input base
			obase=output base
			ibase=2 (i/pin bin o/p in deci)
			obase=2 (i/p in deci o/p in bin)**

6.	Record your session	script <file name> exit script –a	**Start scripting Come out from script Append to old script file**
7.	Use mail oriented commands *** will not work in telnet session**	mail guest1 Subject : Hi HOW R U? ^D CC: guest1 $mail &?	**mail <recipient name> sending mail to guest1 Subject line Message Save & Exit Copy to: displays mail message get help in Mail box**
8.	Change your password	passwd	**Change Password Ctrl-c to cancel**
9.	Display who are the users with various options	who –u who –H whoami	 **With headers Which user logged in**

10.	Know your machine characteristics	uname env	**Displays OS name**
11.	Know your terminal	tty	**Terminal Tele Type (TTY)**
12.	Know your machine characteristics with various options	uname –r uname -n	**Displays release version** **Machine name**
13.	Display and set the terminal characteristics and change the settings	stty stty -echo sty echo	**Setting tty** **Non visible mode, commands given by users are not visible output of the command is visible** **Set back to visible mode**

Note : Internal Variables -> Built-in variables are $PATH, $USER, $HOSTNAME, $HOME etc.

$echo $HOSTNAME	Prints machine name
$echo $USER	Prints user name
$echo $MACHTYPE	Identifies system hardware
$echo $BASH_VERSION	Print shell version
$echo $OSTYPE	Type of OS

Lab 3: File System

Sl No.	Action	Command	Output
1.	Check your current directory	pwd	**Present Working Directory**
2.	change the current directory	cd	**Change Directory**
3.	Making directories	mkdir <dir>	**Directory creation**
4.	Removing directories	rmdir <dir> rmdir –r	**Remove empty directories** **Remove directory recursively**
5.	Use absolute path names to display a file contents	/home/cse/cs3a101 *(Starts from root directory)*	**A pathname that explicitly identifies all directories from the root directory to an individual file.**
6.	Use absolute pathname for a command	/usr/bin/who	**Displays who are logged in**
7.	Use relative pathname to display a file content	*cd ..* *cd ../..* *ls . & ls ..* *(Starts from working directory)*	**One prev directory** **Two prev directroires**
8.	Use . and .. in relative pathnames	. ..	**Working directory** **Parent directory**
9.	List directory contents	ls, ls –R, ls -x	**Listing files, Recursive listing, multi columnar listing to listing directory contents**
10.	Use ls options	ls –l, ls –a,	**Long listing, hidden file listing**
11.	Display a file and create a file	cat <fn> cat > <fn>	
12.	Use cat with various options	cat –n <fn>	**Displays output with line numbers**
13.	Copy a file with various options	cp <src> <des> cp -i	 **Interactive mode**
14.	Delete a file	rm <fn> rm –i rm -f	**Remove filename** **Remove interactive mode** **Remove with force option**
15.	Delete all files	rm *	**Delete All files**
16.	Rename a file	mv <src><dest>	**Move <src> to <dest>**
17.	Paging output of a file and navigate it	more <fn> less <fn> cat <fn>|more cat <fn>|less	**Displays text one screen at a time (move f/w and b/w with f & b)** **Opposite of more command (move f/w and b/w with f & b)** **Does not move f/w and b/w** **move f/w and b/w with f & b**
18.	Repeat factor	2f, 2b	**In more 2f means scrolling forward by 2 pages, 2b- scrolling backward by 2 pages '.' repeat the previous command in more**
19.	Search for pattern in file	grep <pat> fn	**Pattern searching for <pat>**
20.	Use more in pipeline	ls | more	**Page wise listing of ls command**

Lab 4 : Handling ordinary files and basic file attributes

Sl No.	Action	Command	Output
1.	Printing a file with options	lpr <fn> fn1 fn2 lpq lp –t "MYNAME" fn1	Sends file/s to printer Gives status of printing in Q Prints with title string in first page
2.	Cancel the printing	lprm id / owner name or	Remove the print job from the queue
3.	Know the file types **See notes**	file ls > fn file –f <fn>	Determine file type regular,dir,device Takes file names from a file
4.	Count lines, words and characters with various options	***wc –c <fn>,** wc –l <fn> wc –w <fn> wc --version	No. of bytes, No. of lines No. of words Output version info
5.	Display data in octal with various options **See notes for usage**	od <fn> od –bc <fn> od –h <fn>	Dump of a file with octal representation. Displays octal equivalent of each characters Hexadecimal dump
6.	Compare two files	cmp <f1> <f2> cmp –l <f1><f2>	Compare two files byte by byte and return location of first mismatch. Returns nothing if files are same Returns byte and line number at which the first difference occurred is reported Detailed list of byte number & differing bytes in **octal**
7.	Display what are common between two file contents *To drop a particular col, use its col number as prefix with -*	comm <f1> <f2> or comm -2 <f1><f2> comm. -3 <f1><f2> comm. -13 <f1><f2> diff <f1> <f2>	Compare two **sorted files** line by line and writes to standard output lines that are common and lines that are unique in 3 columnar output Writes unique in both the files Writes lines from 2nd column Tells you which lines in one file have to be changed to make the two files identical.
8.	Convert one file to another file	unix2dos <fn> dos2unix <fn> dos2unix –n <dosfn> <unixfn>	Convert linux files to dos format Convert dos files to linux format (Linux contains \n Dos contains \r) Convert and write to the same file Convert the dos file and write output to new unix file

9.	Compress and decompress files with various options **See notes**	* gzip <fn> gzip –l <fn.gz> / <fn> (fn after/before compression) gunzip <fn> or gzip –d <fn> gzip/gunzip –r <dir>	Compress the file by providing extension .gz and ***removes the original file*** Lists size of compressed file Size of uncompressed file Compression ratio Name of uncompressed file To restore original and uncompressed file Compress/Decompress all files in directory <dir>
10.	Create an archive **See notes**	tar –cvf <f.tar> <f1….f10>	Create <f.tar> archive by specifying name of archive with –f, the copy or write operation -c and –v verbose to display the progress and file names as arguments.
11.	Extract files from archive	tar –xvf <f.tar>	-x option Extract files from archive
12.	View the archive	tar –tvf <f.tar>	-t option viewing the Archive
13.	Do compression archiving together **See notes for special feature**	zip <fn.zip> <f1….f10> zip –r <fn.zip> <dir> unzip –r <fn.zip> unzip –v <fn.zip>	Creates fn.zip which combines the compressing function of **gzip** with the archival function of **tar** Recursive compression, compress a directory Files are restored, uses the compressed filename as argument Viewing the archive file

Notes

1. Type field: The first character in the field indicates a file type of one of the following:

    - d = directory
    - l = symbolic link
    - s = socket
    - p = named pipe
    - - = regular file
    - c= character (unbuffered) device file special
    - b=block (buffered) device file special

5. This is helpful to detect any special character and nonprinting characters in your file, or if you want display binary file

9. File compression: File will compressed to a fraction of its original size
File Archive : Group a set of files into a single file.

13.The special feature of zip command is that it doesn’t overwrite an existing compressed files. If <fn.zip> exists, files will either be updated or appended to the archive.

Lab-4 Continued ….

14.	List file attributes **See Notes for total <n>**	ls –l	Long listing with file attributes
15.	List directory attributes	ls –ld <dir1….10>	Directories are identified from the first character of the first column.
16.	Display file permission **See Notes**	ls –l <f1…..f10>	View the permission of few files
17.	Changing file permissions **File Security with permissions**	chmod u+x <fn> chmod u-x <fn>	Change mode set permission u+x for file fn.(+) Removes permission (-)
18.	Use relative permission	chmod ugo+r <fn> chmod a+x <fn>	Set execute permission to user, group, others 'a' for all implies 'ugo'
19.	Use absolute file permission	chmod 777 r=4, w=2, x=1 chmod 000 <fn>	Set permission using octal representation. 777= ugo+rwx Security implication, not able to read, write execute to <fn>
20.	Use chmod recursively **See notes**	chmod –R a+x <dir>	Descend a directory hierarchy and apply the expression to every file and subdirectory it finds.
21.	Use directory permissions **See Notes**	chmod u-r <dir> chmod –w <dir> (without u means by default user) chmod –x <dir>	Listing files ls will not work, permission denied Cannot create files cd <dir> not possible
22.	Change file ownership	chown <owner> <file> chown –R (recursive)	Change ownership of file to specified owner , **it needs super user permission**
23.	Change group ownership **See notes**	chgrp <grpname> <file>	Change group owner ship **(no super user permission required)**

Notes :

14. The ls –l list preceded by the words **total <n>** which indicates total of 'n' blocks are occupied by these files in the disk**,** each block consisting 1024 bytes.
16. r-> read permission, cat can display a file
    w-> write permission, edit such file with editor.
    x->execute permission, the file can be executed as a program
    owner-group-others(world)
18. In relative manner chmod only changes the permissions specified in the command line and leaves the other permission unchanged
21. The file can't be accessed when it has no read permission but can be removed even when it is write-protected.
22. Directory permissions are differs from those of ordinary files.
    The default directory permission are rwxr-xr-x (755). ***The directory must never be writeable by group and others.***
    r-> list of filenames stored in that directory (using ls)
    w->permission to create remove or copy files in the directory
    If we remove write permission the modification of existing file is possible.
    x-> user can enter in to the directory in searching for subdirectories. (cd dir)
23. By default, the group owner of a file is the group to which the owner belongs.

Lab 5: Vi editor

Sl No.	Action	Command	Output
1.	Open vi editor **See Notes**	vi sometext (filename)	Open a file by name sometext
2.	Change to various modes of vi editor **See Notes**	~Command mode(default) i -> Insert mode ESC : (ex mode)	Pass commands to act on text is used to copy and delete text Switch to Insert mode to enter text Revert back to command mode Invoke ex mode
3.	Use repeat factor	10k, 20l	Moves 10 lines up, Moves 20 chars left
4.	Go to Insert mode	ESC i	
5.	Insert text	i or a <text>	
6.	Insert text in line extremes	I and A	I-> Inserts text at beginning of line A-> Appends text at end of line (used to make comment lines in "C")
7.	Opening a new line	O o	O (above) o (below)
8.	Replacing a text	r R S	Replaces a single character with r Replaces all text on the right of the cursor Replaces the entire line irrespective of cursor
9.	Save & Quit (ex mode)	:wq	Writes(save) and quit
10.	Save your work	:w :w <fn>	Save and continue, Like Save as in WINDOWS
11.	Saving and quitting	:x	Save(write) and quit
12.	Aborting the editing	:q :q!	Quit without save (Wont work if buffer is unsaved) Ignores all changes made and quits.
13.	Writing selected lines	:10, 50w <fn> :5w <fn>	Writes 41 lines from 10th line to another file Writes 5th line to another file
14.	Escaping to Unix shell from vi editor	:sh exit	Enter into shell mode Exit from shell and back to vi
15.	Recovering from a crash	:recover vi –r <fn>	When power goes off, leaving work unsaved can be recovered
16.	Do the navigation for movement in the four directions	↑k h←→l ↓j	h,j,k,l to move the cursor in 4 directions
17.	Do the word navigation	b e w (repeat factors can be used)	b-> moves back to beginning of the word e->moves forward to end of word w-> moves forward to beginning of word (B E W- for skipping punctuation)
18.	Moving to line extremes	0 (zero) $ 30|	Beginning of the line End of the line Moves cursor to column 30
19.	Scrolling	ctrl-f, ctrl-b ctrl-d, ctrl-u	Scrolls forward, scrolls backward Scrolls half page forward, half pate backward
20.	Absolute movements	:12 or :12G G gg	Goes to line number 12 Goes to end of file (<ctrl-end> in WINDOWS) To the beginning of the file

Notes

1. **Bill Joy created Vi editor. Bram Moolennar improved it and called as Vim (Vi improved)**
2. There are three modes used in **vi**
   a. Command mode b. Input mode c. ex mode (Last Line mode)

A Few Tips

- Undo – ESC u
- Clearing screen (ctrl+l)
- Don't use CAPS LOCK
- Avoid using the PC navigation key
- Use 'vimtutor' to get help

Vi editor Contd…

Sl No.	Action	Command	Output
1.	Editing text	dd (delete) yy (Yank/copy) p (Put/Paste)	10dd->to delete a 10 lines 5yy -> Copy 5 lines p -> Copy below, P-> copy above
2.	Deleting text	x, 4x	Deletes single character, Deletes 4 characters
3.	Moving text	Same as Question1	
4.	Copying text	Same as Question1 3ye	 copy 3 words (e -> end of word)
5.	Joining lines	J	4J joins following 3 lines with current line
6.	Undoing last editing instructions	u ctrl-r	Undo Redo
7.	Repeating the last command	. (dot)	Used for repeating both Input and Command Mode commands (ctrl+y in MSWINDOWS)
8.	Searching for a pattern	/ and ?	/text <Enter> searches for text forward ?text <Enter> searches for text backward
9.	Repeating the last pattern search	n and N **See notes**	Repeats search in same direction of original search, N reverses the direction.
10.	Do substitution-search and replace	:%s/char/character/g :1,5s/text/texts/g :**.**s/text/texts/g (dot s) :$s/text/texts/g	Here, char is replaced with characters globally thorough out the file Substitutes lines 1 through 5 Only current line, Pattern not found if search fails Only last line

Notes:

9. Repeating the last search after coming out of vi and reopens the file

Lab 6.: The Shell

- **Shell is a agent which sits between user and LINUX System.**
- **The shell is a process that runs when a user logs in and terminates when user logs out.**
- **It is a command interpreter and a programming language rolled into one.**
- **Activities of shell are**
  1. **The shell issues the prompt and waits for you to enter a command**
  2. **Then it scans the command for meta character likes '| > * ' and expands abbreviation (like * in rm *)**
  3. **Then it passes on the command line to the kernel for execution.**
  4. **Waits for command to complete**
  5. **After execution prompt reappears and shell returns to its waiting role to start**

Sl.	Action	Command	Output
1.	Process owned by you	ps	Show which shell is running
2.	Use * and ? for pattern matching with commands	 ls a* echo * ls a?	* is wild card meta character matches any number of chars including none Displays all files starting with a Displays all files in directory ?- wild card, matches single character Displays files with file name of **2** character starting with **a**
3.	matching the Dot	ls .* ls .?* ls .??*	Displays home dir listing Same as above Display hidden files in current dir
4.	Character class ***Frame restrictive patterns***	ls –l a[123] ls –l a[1-3]	Lists file a1 a2 a3 Lists file a1 – a3 (range specification)
5.	Negating the character class	ls –l *.[!co] ls –l *.[!co]*	Matches file names with **single** character ext **but not .c or .o**
6.	Matching totally dissimilar patterns	ls –l *.{c,txt}	Displays files with ext .c and .txt both
7.	Use all the shell wild cards in matching the patterns with egs.	ls *.c cp ???? progs	
8.	Use escape sequence ***\ before wild card to remove its special meaning (escape)***	cat > chap* rm chap* cat > my\ file echo \\	Creates a file with name **chap*** Does not removes chapx,chapy but removes **chap*** file Creates a file with name **my file** Escaping \ it self prints \
9.	use quoting ***The special characters are turned off if any thing within quotes***	echo '\' rm 'chap*' rm "my file" echo "$TERM" echo '$TERM'	Displays \ Removes file chap* Removes file with file name **my file** Interprets as Shell command & execute $TERM->displays terminal name Display $TERM as string
10.	Use escaping echo	echo –e "Hi\nHello"	Hi Hello

Single quotes protect all special character
Double quotes to be interpreted as command substitution and the $ as a variable prefix.

11.	Use standard input file for a command *** See notes**	wc wc < a	Takes input from standard input Takes input form a file name **a**
12.	Use standard output file for a command	wc a > anew cat *.c > c_prgs.txt	The command sends the word count of file **a** to **anew** Use wild card saves all C progs in a single file c_prgs.txt
13.	Use standard error file for a command	cat foo 2>errorfile	The diagnostic o/p has sent to errorfile
14.	Use both standard input and standard output	cat – a cat a – a1	First from std input and then from file a First from file a , then std i/p, then from file a1 *mkdir,cd ,rm,cp- will not use neither std i/p nor std o/p* *ls, pwd,who- uses only std o/p* *lp – uses std i/p* *cat,wc,od,cmp,gzip – uses both std i/p & std o/p*
15.	Use null and terminal teletype files	cmp f1 f2 >/dev/null who >/dev/tty	Output will redirected to a ***special file /dev/null*** Lists current users on std terminal can be accessed by independently by several users without conflict
16.	Use pipes	who | wc –l ls | more	Counts number users logged in, No intermediate file will create.
17.	Create a Tee	who | tee user.txt who | tee /dev/tty | wc -l	Tee saves one copy of who output to file user.txt and writes the other output to std o/p
18.	Do command substitution	echo “date” echo’date’ echo Today\’s date is `date`	Displays string **date** Today’s date is Fri Nov 7 12:34:3
19.	Use shell variables	x=5 echo $x unset x	Assigns value 5 to x Displays 5 x is now undefined
20.	Find out the effects of quoting and escaping	echo “I have \$100” echo ‘ I have $100’ echo “PATH IS $PATH current dir is `pwd`”	Prints I have $100 Prints I have $100 Prints $PATH and pwd *($ is evaluated when it is double quoted)*
21.	Using of shell variables	filename=a cat $filename	Assigns filename to var Displays contents of file **a** which assigned to **filename**
22.	Set the path name	mypath=/home/cse/test 1/a cd $mypath;pwd	Sets path to a variable Change dir to path which is assigned by variable

23.	Use command substitution	mydir = pwd
		mydir=`pwd’
		echo $mydir
			Assigns pwd string to mydir
			Assigns interpreted pwd to mydir
			Displays current working directory

24.	Concatenate variables and strings	base=a ext=.html
		file= $base$ext
		cat $file
			Assigns file name to base and .html to ext
			File contains concatenated string
			Display contents of file

Notes:

11. Standard input sources are
Keyboard, redirection symbol <, using pipeline
How input redirection works
wc < a.txt

a. on seeing the < the shell opens the disk file, a.txt for reading
b. It unplugs the std. input file from default source and assigns to a.txt
c. wc reads from std input which has reassigned by shell to file a.txt

12. Standard output sources are
The terminal, using redirection symbol >, >> and i/p to another prg using pipeline
How output redirection works
wc a.txt > b.txt
a. on seeing the > the shell opens the disk file, b.txt for writing
b. It unplugs the std. output file from default source and assigns to b.txt
c. wc opens the file a.txt for reading
d. wc writes from std output which has reassigned by shell to file b.txt

13. The standard files represented by a number called *file descriptor*
0- Standard Input, 1 – Standard Output , 2- Standard error 3- normal file

The null device is typically used for disposing of unwanted output streams of a process, or as a Convenient empty file for input streams.

Lab 7: The Process

Notes :

A process is an instance of a running program.

The multitasking nature of UNIX allows a process to generate (spawn) on or more process

A file can be treated as a simple file when it is in a disk and we can take it as a process when it is executed.

Sl. No.	Action	Command	Output
1.	Display shell process	echo $$ echo $SHELL	T o know the PID of your current shell Displays Shell's Path Name
2.	use parents and children process	cat <fn> who | wc -l	Parent- Shell (sh,ksh,**bash**,csh) Child – cat command Create two processes 'who' and 'wc –l'
3.	Display process status	ps	
4.	Use ps options	ps –f ps –u <User> ps -a	Displays full listing PID-Process ID, PPID –Parent Process ID, UID-User ID of each process Displays activities of User Lists process of all users except system process
5.	Display system processes	ps –e or –A	Displays the system process generated during system startup
6.	Run jobs in back ground	cat a1>a2& cat yes>y& **yes command**	Returns a number PID, No logging out (we can log out in c Shell & Bash)
7.	log out safely	nohup cat a1& ps -ef	No Hangup, permits execution of the process even after the user logged out. Displays PID and Appending output to nohup.out Display process with full listng
8.	Job execution with low priority	nice cat a1& nice –n 5 cat a1 &	Built-in command, ranges between 1-19, higher nice value implies a lower priority. ***The power to increase priority is reserved for the superuser***
9.	Use kill command	kill <PID> yes > y& kill <PID> kill –s kill <PID> kill $!	Terminates the job having PID <PID> To kill process which is not killed by kill <PID> Kills the Last background job
10.	job control commands	 bg fg jobs fg %1 bg%2	**A job is the name given to a group of processes** A job in background process Bring back to foreground , ^D to kill ^Z to resume Listing status of jobs, Lists priority in [], + most recent job process Brings 1 job to foreground Send 2 job to background
11.	**Execute later commands**		
12.	One time execution commands	at <hh:mm> at>cal >mycal ^D at –l atrm <jobno>	Does not indicate the name of script to be executed. Cal command stored in mycal file at hh:mm Listing of jobs Removing of jobs

13.	Execute in batch queue	batch	Schedules batch of jobs for later execution. Jobs are executed as soon as the system load permits.
14.	Run jobs periodically **See Notes**	crontab –e crontab -r	crontab edit, opens editor like vi to remove crontab file
15.	Create crontab file **See Notes**		00-10 17 21 11 5 ls –l *.c >mycfiles (1) (2) (3)(4)(5) (6)
16.	Timing the processes	time	time who time ls –l real -> clock elapsed time from invocation of the command until its termination user->time spent by the program in executing itself sys->time used by the kernel

Notes :

6. Yes [String]- Output a string repeatedly until it is killed
14. ps –e shows the cron.d daemon running. This is LINUX system's chronograph, ticking away every minute. This executes programs at regular intervals.

15. 1. Execution every minute in the first 10 minutes of the hour (00-59)
2. 17-> 5 pm (24 hours clock)
3. day of the month (1 to 31)
4. specifies month (1-12)
5. days of the week (5-Friday , 0-6)
6. Command to be executed every minute in the first 10 minutes after 5 pm every Friday of the month November (of every year)
7. A * is used in any of the first five fields implies that command is to be executed every period depending on the field where it is placed.

Eg. Cron tab entry to execute myscript.sh file every 30 minutes on every Wednesday and Friday between the times 8am and 5 pm is

00-30 8,17 * * 3,5 myscript.sh

Lab 8: Customizing the environment

There are two type of shell variables, local variables and environment variables
PATH,HOME, & SHELL are environment variables which are available in the user's total environment, the sub shell

Sl. No.	Action	Command	Output
1.	Display all the environment variables **See Notes**	echo $PATH echo $HOME env $x=10 ; echo $x sh echo $x	Displays path Displays Home directory Displays only environment variables Assigns/displays value to local variable x=10 Invoke child shell Will not print any thing as x is a local variable but echo $PATH will work

2.	Display variables used in bash and korn shells. Change them also.	PS1='[$PWD] '
		
		PS1='[\!] '
		PS1="\h> "
		PS1="Hi xxx> "
			Changes the primary prompt to display current directory.
			Sets prompts to current event number
			Sets prompt to hostname
			Sets prompt to Hi xxx>

3.	Use aliases **(Short hand names to frequently used command)**	alias myll = 'ls –l' alias mycd = "cd /home/cse/sri/a" alias cp="cp –i" unalias cp \cp	Alias myll defined for the command ls –l Alias mycd defined for change dir to a Alias cp defined to modify cp to interactive Unset alias To run original cp command
4.	Maintain history file	history history 5	Displays history list showing the event number Displays last 5 commands executed
5.	Access previous commands by event numbers and context	!! !38 !38:p !v !h	Repeat previous command Repeat the command by event/cmd number (38) Print the 38th command without execution Repeat the previous command which starts with letter 'v' or 'h'
6.	Substitution in previous commands	ls –l *.c !ls:s/c/txt	Lists file which .c extension Repeat same command which replace *.c with *.txt -> displays all files with .txt extension
7.	Use last argument to pervious command	mkdir newdir cd $_	$_- Last argument to previous command (newdir) Change to dir <newdir>
8.	Use history variables	vi .bash_history HISTSIZE=100	View History file Size of history list set to 100
9.	Use inline command editing	set –o vi set +o vi	Provides vi-like capability of editing the cmd line (built –in setting in bash) Revert back to non vi mode
10.	Search a history for a command	:<ESC>/ls :<ESC>/ps	Locates last occurrence of string ls Locates last occurrence of string echo
11.	Use set –o	set –o noclobber cat >a1 set –o ignoreeof set +o ignoreeof	Avoids accidental overwriting If file <a1> already exists displays error msg Avoids accidental logout if we press ^D ^D to logout

12.	Use tilde substitution	~ cd ~/a cd ~- (hyphen)	Shorthand representation of the home directory Change directory $HOME/a Change to previous directory /toggles
13.	Initialization scripts	ls –a	Displays all hidden files
14.	Use profile	vi .bash_profile	PS1='[$PWD] ' echo "Today's date is : `date`" When user logs in the prompt will changed and Today's date is : <date> will print. These commands are executed only once in a session.
15.	Use the rc file	vi .bashrc . ~/.bashrc	The file is executed every time a second shell is called up The rc file will be executed after the profile To run bashrc file (Aliases history setting & set –o commands are kept in rc file)

Notes

1. Internal variables/Environment variables, Built-in variables

Common environment variable are

$echo $PATH $echo $HOME $echo $MAIL	$echo $SHELL $echo $TERM $echo $LOGNAME	$echo $PS1 $echo $PS2

Lab 9. More file attributes

Sl No.	Action	Command	Output
1.	Create hard links **See Notes**	ln <srcfn> <desfn> ls –il	Works like cp command, (shortcut key in win) Lists file with inode number
2.	Remove hard link	rm <fn>	Removes file by deleting its directory entry to remove link
3.	Use hard link	ls –il /bin/gzip /bin/gunzip	gzip and gunzip are hard linked files having same inode numbers
4.	Create symbolic /soft link **(see notes)**	ln –s <srcfn> <desfn>	Provides the pathname of the file that actually has the contents, displays lengtn of the pathname it contains (srcfn)
5.	Use links for directory with permissions(RWX)	ln –s <srdirnm> <desdirname>	
6.	Use umask	umask	022, files -> 644(666-022) dir->755 (777-022) (022 is system wide default setting) ***(cannot be changed by any one even su, use chmod when it is required)***
7.	Retrieve modification and access time **(Notes)**	ls –l <fn> ls –lu <fn> ls –lc <fn> ls –lt ls -lut	Time of last file modification Time of last access Time of last inode modification Displays listing in order of their mod’fn time Displays listing in order of their access time
8.	Change the time stamps ***cat > <fn>*** ***ls –lt <fn> (after 1 min)*** ***touch <fn>*** ***ls –lt <fn>***	touch <fn> yyyymmddhhmm touch –t 200810201443 <fn>	Set the modification and access times to predefined values, creates <fn> with current time. Creates <fn> with date 20-10-2008 with time 14:43 (elapsed time)
9.	Locate files	find / -name <fn> -print find . –name “*.c” -print	/ - search starts from root directory .- search starts from current directory for .c files
10.	Use selection criteria	find . –inum <no> -print find . –type d -print	Find all filename that have the <no> Type f (ordinary file) d(dir) l (sym link)
11.	Use find operators with options	find . ! –name “*.c” –print find . \(-name “*.win” –o –name “*.l” \) -print	Displays all files other than .c files (not)

Notes:

2. Why Hard links
   - Protects against accidental deletion when they exists in different directories
   - We don’t need to maintain two programs as two separate disk files if there is a small difference between them.

   Many LINUX commands are linked.
   Ex : ls –il /bin/gzip /bin/gunzip
   Limitations of Hard Links
   - You can’t link a filename in the /usr file system to another in the /home file system
   - You can’t link a directory even within the same file system.

4. The files are not identical when we create soft links. The <desfn> linked file can be deleted and recreate the link. But if we remove <srcfn> we would lose the file containing the data and the link point to nonexistent file and become a *dangling* symbolic link.

7. Whenever you write to a file, the time of last modification is updated in the file’s inode.
   A directory can be modified by creating, removing, and renaming files in the directory.
   Changing file contents changes last modification time not its directory.

File Access time : is the last time someone read, wrote or executed the file and is distinctly different from modification time. Creating or removing a file or doing “cd” to a directory doesn’t change its access time.

Lab 10.Simple Filters

Sl No.	Action	Command	Output
	Create list for info maintenance *		
1.	Use pr with options	pr emp.lst cat num | pr –t -5 pr –t –n –d –o 10 emp.lst **try other options**	Prepares emp.lst file for printing with headers Prints file num without headers and 5 col output -d –double space, -n –line numbers, -o n increases left margin to 10
2.	Display the beginning and end of a file with options	head –n 2 emp.lst vi \`ls –t | head –n 1\` tail –n 3 emp.lst tail –c -13 emp.lst tail –c +13 emp.lst	Displays first 2 lines from beginning of a file (default is 10 lines without –n option) Opens last modified file for editing Displays last 2 lines from end of a file Copies last 13 bytes from emp.lst Copies everything after skipping 12 bytes
3.	**Slit a file vertically**		
4.	Cutting columns and fields	cut –c 6-22,24-32 emp.lst cut –d \\\| -f 2,3 emp.lst **or** cut –d “|” -f 2,3 emp.lst who | cut –d “ “ –f1	Extract 6-22 & 24-32 column data Extract 2nd and 3rd col with | as delimiter from emp.lst Extracting user list from who output
5.	Pasting files with option	paste emp.lst emp1.lst paste –s –d “||\n” addr_book	View two file side by side by pasting them. Joining lines (|| for two column separator)
6.	Ordering a file with options	sort a1 sort –t “|” –k 2 emp.lst sort –t “|” –r –k 2 emp.lst sort –t “|” –k 4,4 –k 2,2 emp.lst sort –t “|” –k 5.9,5.10 emp.lst sort num -> sort –n num cut –d”|” –f3 emp.lst | sort –u | emp.lst sort –o emp_asc –t “|” –k 2 emp.lst sort –c emp.asc	Sort the contents of a1 (contains 1 column) Sort on 2nd field (name) Sort in reverse order Sort on more than one key (dept,name) Sort on character position (joining date) Numeric sort -u for unique(removing repeated lines) Sorted output (-o) stored in emp_asc file Check for sorted file
7.	locate repeated and non repeated lines with options	uniq dept.lst uniq –u dept.lst uinq –c dept.lst	Fetches one copy of each line and writes Fetches non-repeated lines (-u unique) Counting frequency of occurrence (-c count)
8.	translating characters with options	tr ‘|’ ‘~’ < emp.lst tr ‘|/’ ‘~-‘ < emp.lst tr ‘[a-z]’ ‘[A-Z]’ < emp.lst tr –d “|” < emp.lst tr –s ‘ ‘ < emp.lst	Replace all ‘|’ with ‘~’ Replace all ‘|’ and‘~’ with ‘/’ and ‘-‘ Changing case of text Delete character ‘|’ and writes Suppress ‘ ‘ and writes.

*** 1. emp.lst**

```
1001|Asha   |director  |Sales       |03/09/2008|6700
1002|Usha   |director  |marketing   |26/09/2007|8207
1003|Nisha  |director  |production  |12/02/2008|7000
1004|Harsha|Director  |Sales       |12/12/2007|7807
2005|Pasha  |Manager |Sales       |09/08/2008|9000
```

dept. lst

```
accounts
accounts
admin
marketing
marketing
personnel
production
sales
sales
```

Lab 11:Filter using Regular Expressions

Sl.	Action	Command	Output
1.	Search for pattern with options grep- Global Regular Expression Print	grep “sales” emp.lst who | grep <uname> grep “director” emp.lst emp1.lst grep –i “director” emp.lst grep –v “director” emp.lst grep –n “director” emp.lst grep –c “director” emp.lst grep –c “director” *.lst grep –l ‘manager’ *.lst grep –e –i “manager” –e “marketing” *.lst grep –f pat.lst emp1.lst	Search for sales in emp.lst Search for uname logged in Displays file along with output Ignore case – searches for Director Deleting lines (inverse of the above) Displaying line numbers Counts lines containing pattern Counts lines containing pattern in *.lst files Displays file names contains pattern Matching multiple patterns-Displays data contains manager & marketing Taking pattern “Manager” from pat.lst
2.	***BRE (Basic Regular Expression)*** Use *,.,^,$,literals(constants),+,?,-E,|,(,) for searching a pattern ***grep –E or egrep Extended Regular Expression (ERE)***	grep “[dD]irector” emp.lst grep “mark*” emp.lst (mar*) grep “3….” emp.lst grep “^1” emp.lst ***Pattern locations*** grep “^[^1]” studlist” ls –l | grep “^d” grep “7…$” emp.lst grep “7$” emp.lst grep –E “K?[nm]al” emp1.lst grep –E “Asha|Usha” emp.lst grep –E “(Kun|Kom)al” emp1.lst	Matches D/director (d or D) Matches immediately preceding char Matches single char starting from 3 Matches at the beginning of the line starts from 1 Negation. Shows only directories Matches at the end of the line ? Matches zero or more occurrence + Matches one or more occurrence (try) Locate for Asha and Usha ( ) for group patterns locate for Kunal and Komal
3.	Use the stream editor	**SED-Steam Editor- performs non interactive operations- Line editor- Learning sed will prepare well for perl which uses many of these features.** **Syntax : sed options ‘address action’ <fn/s>**	
4.	Use line addressing in sed	sed ‘3q’ emp.lst ***try sed ‘$q’ <fn>*** sed –n “1,3p” emp.lst sed –n “$p” emp.lst sed –n “1,2p ↵ 4p↵ ‘ emp.lst ↵ sed –n ‘3!p’ emp.lst ***1,3!p***	**Q**uits after 3rd line **P**rints 1 to 3 lines -n is must with p cmd Prints Last line Selecting multiple group of lines Selecting 1,2,4 the lines and print !-Negation, Doesn’t print 3rd line
5.	Use multiple instructions in sed	sed –n –e ‘1,2p’ –e ‘$p’ emp.lst sed –n –f <ins_fn> emp.lst	Prints 1st, 2nd, Last line form emp.lst Take instruction from a file (try)
6.	use context addressing in sed	sed –n ‘/[dD]irector’ emp.lst sed –n ‘/Usha/,/Nisha/p’ emp.lst sed –n ‘/07.....$/p’ emp.lst sed -n ‘/mark*/p’ emp.lst	Locates for D/director and print Locates for Usha till it finds Nisha Locates of 2007 join date *5 dots for |**2000** after 07 in 2007* Matches immediately preceding char
7.	Write selected lines to a file	sed –n ‘/director/w dlist’ emp.lst sed –n ‘1,2w mylist’ emp.lst	Write the selected line to separate file Saves 1st and 2nd line to mylist file

8.	Insert, change and delete lines using sed	sed ‘2i\Inserting a sentence in 2nd line\’ emp.lst > $$ cat $$ sed ‘a\↲ >↲ ***try*** >’ emp lst >semp.lst sed ‘/director/d’ emp.lst > $$ sed ‘/^[□⇆]*$/d’ semp.lst > $$	**I**nserts a sentence given to 2nd line and redirect output to a temp file $$ Displays contents of $$ after insertion. **A**ppends a blank line after each line and prints on terminal **(try)** **D**eletes lines contains directory saves in $$ Delete blank lines.A space and tab inside[ ]
9.	Use substitution in sed	sed ‘s/|/:/’ emp.lst sed ‘s/|/:/g’ emp.lst sed ‘1,3s/|/:/g emp.lst	Replace | with : for first col g for whole file replaces for 1 to 3 lines (first 3 lines) try for *,^,$, .
10.	Use remembered pattern, repeated pattern	sed ‘s/director/member/’ emp.lst sed ‘s/director/executive &/’ emp.lst	Search for director and replaces with member (it remembered pattern ***director***) Replace ***director*** with ***executive director*** (source pattern occurs in destination)
11.	Use IRE	cat emp2.lst grep ‘[0-9]\{10\}’ emp2.lst	Search for digits(0-9) which contain 10 digits
12.	Use TRE	Tagged Regular Expression	

emp.lst

```
1001|Asha  |director|Sales       |03/09/2008|6700
1002|Usha  |director|marketing   |26/09/2007|8207
1003|Nisha |director|production  |12/02/2008|7000
1004|Harsha|Director|Sales       |12/12/2007|7807
2005|Pasha |Manager |Sales       |09/08/2008|9000
```

emp1.lst

```
1001|Anil  |Manager  |Sales       |03/09/2008|6700
1002|Sunil |Executive|marketing   |26/09/2008|8200
1002|Suneel|Executive|marketing   |26/09/2008|8200
1003|Kunal |director |production  |12/02/2008|7000
1004|Komal |director |Sales       |12/12/2008|7800
```

emp2.lst

```
1001|Anil  |Manager  |Sales       |03/09/2008|6700|9845099234
1002|Sunil |Executive|marketing   |26/09/2008|8200|9876712348
1002|Suneel|Executive|marketing   |26/09/2008|8200|9845173213
1003|Kunal |director |production  |12/02/2008|7000|98653
1004|Komal |director |Sales       |12/12/2008|7800|
```

BRE – Basic Regular Expression : *, [pat], ^, $
ERE – Extend Regular Expression : uses –E option : + ? | ( )
IRE – Interval Regular Expression
TRE – Tagged Regular Expression