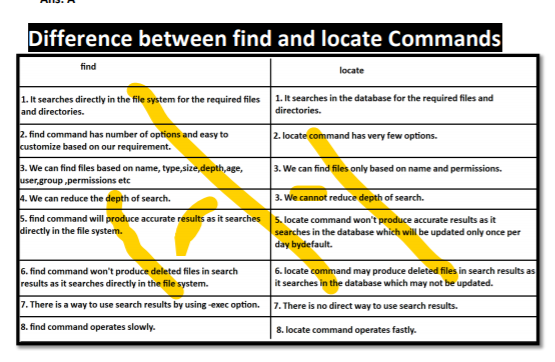
Linux Commands:

* Linux - It is an operating system, by using that, users/applications can communicate with hard ware components
* Shell - It reads our command, verify syntax and check whether the corresponding command related application is available or not and converts to kernel understandable form
* Kernel - It is responsible to interact with hardware components.and Memory allocation and processor allocation will takes care by kernel
* Linux has 3 file types – normal files, Directory files and Device Files
* The first character represents the type of file. **d** Directory File, - Normal File, **l** Link File c Character Special Fil,e **b** Block Special File **s** Socket File
* cd ~/cd Changes to user home directory
* cd - Changes to previous working directory
* Linux Folder Structure
* Bin - bin contains binary executables related to commands used by normal user
* Sbin - sbin contains binary executables related to commands used by superuser.
* Etc - This directory contains all system configuration file
* tmp means temporary. It contains all temporary files created in the current session
* dev means device. In Linux, everything is treated as a file including devices also. i.e every device is represented as a file. By using these files, we can communicate with the devices
* mnt - We have to attach external file system files from Pen drive, CD, external hard disk etc to the Linux File System. Then only we can use those external files. This attachment process is called mounting. In the old operating systems, we have to perform mounting manually. But in recent operating systems, mounting is performing automatically and we are not required to perform manually. • The files of manual mounting will be placed inside mnt directory
* media - The files of automatic mounting will be placed inside media directory
* mnt ◊ Contains manual mounting files.
* media ◊ Contains automatic mounting files
* opt - This directory contains all 3rd party software installation files
* lib means library. It contains Linux os libraries which are required by our commands and applications
* var means variable data. If any data which is keep on changing, such type of data will be stored inside var directory. • log files will be stored inside var.
* home - As linux is multi user operating system, for every user a separate directory will be created to hold his specific data
* • proc means processes.
* Root It is the home directory of super user.
* Boot This directory contains the files which are required to boot linux os.
* usr means user. This directory contains all user related softwares.
* Ls - It will display all files and directories according to alphabetical order of names.
* Ls –r It will display all files and directories in reverse of aplhabetical order.
* ls | more To display content line by line
* ls –l To display long listing of files
* ls –t To display all files based on last modified date and time. Most recent is at top and old are at bottom.
* Ls –a To display all files including hidden files. Here . and .. also will be displayed.
* Ls –A To display all files including hidden files except . and ..
* Ls –i To display all files including inode number.
* ls –R It will list all files and directories including sub directory contents also
* ls –s The number of blocks used by file will be displayed 1 Block = 1Kb
* ls –h display in human readable format
* $ ls /dev | less, ls /dev | more
* ls /dev | head -5 , ls /dev | tail -5
* To display only date in the form: mm/dd/yy - date +%D
* To display only day value - date +%d
* To display only year value in yy form - date +%y
* $ cal To display current month calendar. cal 2020 - To display total year calendar.
* $ cal 08 2019 ◊ To display august 2019th calendar
* mkdir dir1, mkdir dir1 dir2 dir3
* mkdir dir1/dir2/dir - To create dir3. But make sure dir1 and in that dir2 should be available already.
* mkdir -p dir1/dir2/dir3, -p means complete path
* mkdir dir{6..10} $ touch dir{6..10}/{a..d}.tx
* rmdir command will work only for empty directories. If the directory is not empty then we will get error. We cannot use rmdir for files. Hence the most useless (waste) command in linux is rmdir.
* Whenever we are using rm command for directories, we should use -r or -R option. Here case is not important.
* Rm –I, if we want confirmation then we have to use –i
* While removing files and directories, if we don't want any error messages, then we should use -f option. It is opposite to -i option.
* If we want to know the sequence of removals on the screen we should go for -v option.
* $ cp source\_file destination\_file
* $ cp file1 file2 output - file1 and file2 will be copied to output directory. Here we can specify any number of files, but last argument should be directory.
* To Copy all Files of One Directory to another Directory: cp dir1/\* dir2
* cp dir1 dir2
* Whenever we are copying one directory to another directory, compulsory we should use -r option
* cp -r dir1 dir2 - total dir1 will be copied to dir2
* $ cp -r dir1 dir2 dir3 dir4 dir5 • dir1,dir2,dir3 and dir4 will be copied to dir5
* Renaming of files: $ mv oldname newname Eg: $ file1.txt file2.txt file1.txt will be renamed to file2.tx
* Renaming of Directories: $ mv dir1 dir2 dir1 will be renamed to dir2
* Moving files to directory: $ mv a.txt b.txt c.txt output a.txt,b.txt and c.txt will be moved to output directory
* Moving of all files from one directory to another directory: $ mv dir1/\* dir2 All files of dir1 will be moved to dir2. After executing this command dir1 will become empty
* Moving total directory to another directory: $ mv dir1 dir2
* Note: If dir2 is already available then dir1 will be moved to dir2. If dir1 is not already available then dir1 will be renamed to dir2
* Creation of Files: In Linux, we can create files in the following ways: 1) By using touch command (to create empty file) 2) By using cat command 3) By using editors like gedit, vi, nano et
* Eg: $ cat > file1.txt Hello Friends Listen Carefully Otherwise Linux will give Left and Right ctrl+d ◊ To save and exit
* touch for creating empty file where as cat for creating a file with some content
* > meant for overwriting >> meant for appending/concatenation
* ) If we are using Touch Comamnd, but the File is already available then what will happend? The content of the file won't be changed. But last modified date and time (i.e., timestamp) will be updated.
* We can view content of the file by using the following commands 1) cat 2) tac 3) rev 4) head 5) tail 6) less 7) more
* View Content of the File by using cat Command $ cat file1.txt
* While viewing file content we can include line numbers by using -n option.
* $ cat -n file1.txt
* Copy content of one file to another file $ cat input.txt > output.txt 5) To copy content of multiple files to a single file $ cat file1.txt file2.txt file3.txt > file4.txt 6) Merging/appending of one file content to another file $ cat file1.txt >> file2.txt
* 2. tac Command: It is the reverse of cat. It will display file content in reverse order of lines. i.e first line will become last line and last line will become first line
* rev means reverse. Here each line content will be reversed. It is horizontal reversal.
* cat command will display total file content at a time. It is best suitable for small files. If the file contains huge lines then it is not recommended to use cat command. We should go for head, tail, less and more commands.
* head Command: We can use head command to view top few lines of content
* head -n 30 file1.txt OR head -30 file1.txt • To display top 30 lines of the file. • Instead of 30 we can specify any number.
* head -n -20 file1.txt To display all lines of file1.txt except last 20 lines.
* head -c 100 file1.txt To display first 100 bytes of file content.
* . tail Command: • We can use tail command to view few lines from bottom of the file. • It is opposite to head command
* tail -n 30 file1.txt OR tail -30 file1.txt OR tail -n -30 file1.txt It will display last 30 lines. ✽ tail -n +4 file1.txt It will display from 4th line to last line
* more Command: We can use more command to view file content page by page. more file1.txt • It will display first page. • Enter ◊ To view next line • Space Bar ◊ To view next page • q ◊ To quit/exit
* . less Command: • By using more command, we can view file content page by page only in forward direction. • If we want to move either in forward direction or in backward direction then we should go for less command.
* less file1.txt It will display first page d ◊ To go to next page.(d means down) b ◊ To go to previous page. (b means backward)
* vimdiff Command: • It will highlight differences in vim. • To support this command, we have to install vim by using the following command. • sudo apt install vim • vimdiff a.txt b.txt • ctrl+w+w ◊ To go to next window • :q ◊ Close current window • :qa ◊ Close all windows • :qa! ◊ Close all windows forcely.
* There are 2 types of link files 1) Hard Link files 2) Soft Link files. Hard Link - It is just another name of the same exact file. We can create hard link file by using ln command. ln originalfile hardlinkfile
* Eg: ln file1.txt file2.txt Here file1.txt is original file and file2.txt is hard link file
* Important conclusions about hard link file: 1) Both original file and hardlink file have same inode number, same size, same timestamp. 2) If we delete original file, then there is no effect on hardlink file
* 2) Soft Link File: • A softlink is a pointer to another file. It is just like windows shortcut. • It is also known as symbolic link. • We can create soft link file by using ln command but with -s option. • ln -s originalfile softlinkfile • Eg: ln -s file1.txt file2.txt • Here file1.txt is original file and file2.txt is link file.
* ) Original file and softlink file have different inode numbers, different file sizes and different timestamps. 2) Usually softlink file has smaller file size than original file size. 3) If we delete original file then softlink files will become useless.
* Link files for directories: We cannot create hardlink for directories because it breaks Linux File System. Having two root directories is meaningless.
* Note: For files we can create both hard and soft links. But for directories we can create only softlinks butnot hardlinks.
* We can use wc command to count number of lines, words and characters present in the given file. wc filename no\_of\_lines no\_of\_words no\_of\_characters filename
* Eg: $ wc a.txt 4 26 166 a.txt 4 ◊ Number of Lines 26 ◊Number of words 166 ◊ Number of characters (File size in bytes) We can use the following options with wc Command -l ◊ To print only number of lines -w ◊ To print only number of words -c ◊ To print only number of characters -lw ◊ To print only number of lines and words -lc ◊ To print only number of lines and characters -wc ◊ To print only number of words and characters -L ◊ To print number of characters present in Longest Line.
* We can sort data of the file by using sort command
* sort filename Here sorting is based on alphabetical order. If the file contains alphanumeric data, then first numbers will be considered and then alphabet symbols.
* If the file contains only numbers, then the sorting is not based on numeric value and it is just based on digits.
* If we want to sort based on numeric value then we have to use -n option. -n means numeric value
* Bydefault sort command will display duplicate lines. If we want only unique lines then we have to use -u option. -u meant for unique lines
* Without using -r option with sort command, sort the content based on reverse of alphabetical order and store the result inside sorted.txt? sort a.txt | tac > sorted.txt sort -r a.txt > sorted.txt
* -k means KEYDEF (key definition). Based on which key (column) we have to sort. $ ls -l /etc | head -10
* Sort based on File Size in ascending Order: $ ls -l /etc | head -10 | sort -k 5n
* We can use uniq command to display unique content in the file. But to use uniq command, compulsory the file should be sorted, otherwise it won't work properly. $ sort a.txt | uniq
* With uniq command we can use multiple options: -d ◊ To display only duplicate lines -c ◊ To display number of occurrences of each line -i ◊ Ignore case while comparing -u ◊ To display only unique lines i.e the lines which are not duplicated.
* Redirecting Standard Ouput: We can redirect standard output by using > and >> symbols. > will perform overwriting of existing data >> will perform appending to existing data $cat 1> output.txt
* Redirecting Standard Error: We can redirect error messages from the terminal to our own file by using > and >> symbols. $ cal 34 w3892384208342 2>> error.txt Now error message won't be displayed to the console and will written to error.txt. For error redirection 2 is mandatory
* Redirecting Standard Input: We can redirect standard input from keyboard to our required file. We can perform input redirection by using < symbol
* \*\*\*Note: To redirect both standard output and standard error to the same destination we can use shortcut as follows $ cat < a.txt &> output.txt &> means both standard output and standard error.
* < symbol meant for input redirection > symbol meant for output redirection where the existing data will be overwritten. >> symbol meant for output redirection where the data will be appended instead of overwriting
* Sometimes we can use output of one command as input to another command. This concept is called piping. By using piping, multiple commands will work together to fulfill our requirement. $ ls -l /etc | wc
* tee Command: Requirement: The output of the ls command should be saved to output.txt and should be provided as input to wc command: ls -l 1>output.txt | wc This command won't work because if we are using redirection in the middle of piping, it will break piping concept. In piping, if we want to save the output of one command to a file and if we want to pass that output as input to next command simultaneously, then we should go for tee command.
* tee command is just like T-Junction or T-Pipe. It will take one input but provides two outputs. Eg 1: To save the output of ls command to a file and to display to the terminal simultaneously. $ ls –l ◊ It will display to the terminal $ ls -l > abc.txt ◊ It will save to the abc.txt but won't display to the terminal. $ ls -l | tee abc.txt
* xargs Command: Q1) Display the Output of Date Command by using echo Command with Piping Concept? $ date | echo ◊ It won't work because the output of date command is stream, but echo command will accept only command line arguments but not stream. $ date | xargs echo ◊ xargs command will convert the output stream of date command into command line arguments and these arguments will be passed as input to echo command. Hence the job of xargs command is to convert output stream into command line arguments Eg 1: Assume input.txt contains file names. Each file contains some data. Read file names from the input.txt, write total content to output.txt and display the total number of lines present in output.txt. $ cat input.txt | xargs cat | tee output.txt | wc -l Eg 2: Assume input.txt contains file names. Read file names from the input.txt and remove all these files. $ cat input.txt | xargs rm
* We can execute multiple independent commands in a single line by using the following two ways 1 st Way: By using semicolon (;) cmd1;cmd2;cmd3;.....;cmdn First cmd1 will be executed and then cmd2 followed by rest of the commands. If any command fails in the middle, still rest of the commands will be executed. 2 nd Way: By using && cmd1 && cmd2 && cmd3 &&..... && cmdn First cmd1 will be executed and then cmd2 followed by rest of the commands. If any command fails in the middle, then rest of the commands won't be executed.
* 1) ◊ Represents zero or more characters 2) ? ◊ Represents only one character 3) []◊ Range of characters 4) [abc] ◊ Either a or b or c 5) [!abc] ◊Any character except a,b and c 6) [a-z] ◊ Any lower case alphabet symbol 7) [A-Z] ◊ Any upper case alphabet symbol 8) [a-zA-Z] ◊ Any alphabet symbol 9) [0-9] ◊ Any digit from 0 to 9 10) [a-zA-Z0-9] ◊ Any alphanumeric character 11) [!a-zA-Z0-9] ◊ Except alpha numeric character (i.e special symbol) 12) [[:lower:]] ◊ Any lower case alphabet symbol 13) [[:upper:]] ◊ Any upper case alphabet symbol 14) [[:alpha:]] ◊ Any alphabet symbol 15) [[:digit:]] ◊ Any digit from 0 to 9 16) [[:alnum:]] ◊ Any alpha numeric character 17) [![:digit:]] ◊ Any character except digit 18){} ◊ List of files with comma separator
* To list out all files present in current working directory ⎝ $ ls \* 2) To list out all files with some extension ⎝ $ ls \*.\* 3) To list out all files starts with a ⎝ $ ls a\* 4) To list out all files starts with a and ends with t ⎝ $ ls a\*t 5) To list out all .java files ⎝ $ ls \*.java 6) To list out all files where file name contains only 2 characters and first character should be 'a' ⎝ $ ls a?
* To list out all files where file name contains only 3 characters ⎝ $ ls ??? 8) To list out all files where file name contains atleast 3 characters ⎝ $ ls ???\* 9) To list out all files where file name starts with a or b or c ⎝ $ ls [abc]\* 10) To list out all files where file name should not starts with a, b and c ⎝ $ ls [!abc]\* 11) To list out all files starts with lower case aphabet symbol $ ls [a-z]\* OR $ls [[:lower:]]\* 12) To list out all files starts with upper case aphabet symbol $ ls [A-Z]\* OR $ls [[:upper:]]\* 13) To list out all files starts with digit. $ ls [0-9]\* OR $ls [[:digit:]]\* 14) To list out all files where first letter should be upper case alphabet symbol, second letter should be digit and third letter should be lower case alphabet symbol. $ ls [[:upper:]][[:digit:]][[:lower:]] 15) To list out all files starts with special symbol $ls [![:alnum:]]\* 16) To list out all files with .java and .py extension $ ls {\*.java, \*.py} Note: We can use these wildcard characters with the following commands also. cp, mv, rm 17) To copy all files starts with digit to dir1 directory. $cp [[:digit:]]\* dir1 $cp [0-9]\* dir1 18) To move all files starts with alphabet symbol and with .txt extension to dir2 directory? $mv [[:alpha:]]\*.txt dir2 19) Remove all files starts with a or b or c and ends with e or t. $rm [abc]\*[et]
* How to Create Alias Names? $ alias nickname='original command' $ alias nickname="original command" After aliasname space is not allowed. Hence the following are invalid alias nickname ='original command' alias nickname= 'original command' alias nickname = 'original command' How to List all available Aliases? By using alias command without any arguments $alias How to Remove Alias Names? By using unalias command. $unalias alias\_name
* Eg: To list out all files present in current working directory, save this data to output.txt and display the number of lines to the terminal. Define alias name 'current' for this total activity. alias current='ls -l | tee output.txt | wc -l' 2. To use other operating system (like windows) commands directly in linux alias cls='clear' alias rename='mv' 3. To handle typing mistakes alias grpe='grep' 4. To handle language barriers: In Germany datum means date. alias datum='date'
* How to persist aliases permanently? Whatever aliases we created, are bydefault available only in the current session. Once we close the terminal, all aliases will be lost. But we can make our created aliases permanently in our system by using the following 2 ways: 1 st Way: We have to define our aliases in .bashrc file present in our home directory. gedit .bashrc Add the following lines in that file. # myown aliases alias cls='clear' alias ddd='date;date;date' Note: To reflect these aliases, compulsory we have to close and open terminal. 2 nd Way: Instead of editing .bashrc file, we can create our own file to maintain our defined aliases. The name of the file should be .bash\_aliases and should be present in home directory
* We can use locate command to locate files and directories in our system. Internally locate command will search in the database for the required files and directories and returns the results. As locate command is searching in the database instead of filesystem, performance will be improved. 1. To locate all .jpg files $ locate \*.jpg
* Reason: locate command using database to find results. This database will be updated only once per day bydefault. After updating database, some files may be deleted. Hence before printing results, to check whether files are existing or not, we have to use -e or --existing option.
* How to Update Database? We can see the database by using locate command with -S option. $ locate –S $ sudo updated
* find Command: We can use find command to find files and directories present in our system. It provides more search options when compared with locate command like 1) Search only files 2) Search only directories 3) Search by name 4) Search by size 5) We can use search results automatically for some other commands 6) We can restrict maxdepth etc
* 1. $find It will find all files and directories in current working directory and below in linux file system. This is the default behaviour.
* 3. maxdepth Option: usually find command will search in all depth levels. But we can specify the required depth level by using maxdepth option 1. $ find . -maxdepth 1
* Find by Type: We can find only files or only directories by using type option. -type f ◊ means only files -type d ◊ means only directories durgasoft@durgasoft-VirtualBox:~/Desktop$ find -type f
* We can find files and directories by name by using -name option. $ touch {A..D}.txt $ touch {A,B}{A,B}.txt
* Find Files by Size: We should use -size option. + symbol means greater than (over) - symbol means less than 1. To list out all file names where size is over 200kb $ find / -type f -size +200k This command required root privileges $ sudo find / -type f -size +200k | wc -l 2. To list out all file names where size is over 200kb but less than 4MB. $ find / -type f -size +200k -size -4M | wc -l 3. To list out all file names where file size is less 200kb or more than 4MB. $ find / -type f -size -200k -o -size +4M | wc –l
* Note: +n for greater than n -n for less than n n for exactly n
* To find all Files and Directories inside /dev Directory upto Maximum of 3 Levels Deep and Size is Greater than 200 Kilo Bytes? C) find / -type f -name '\*.txt'



* As the part of admin activity, it is very common requirement to pack and compress a group of files. The main advanatages are: 1) It improves memory utilization 2) Transportation will become very easy 3) It reduces download times etc This process involves the following 2 activities: 1) Creation of Archive file 2) Apply compression algorithms on that archive file
* )Creation of Archive File We can group multiple files and directories into a single archive file by using tar command. tar ◊ tape archive A) To create tar file tar -cvf demo.tar file1.txt file2.txt file3.txt tar -cvf demo.tar \* B) To display table of contents of tar file tar -tvf demo.tar C) To Extract contents of tar file tar -xvf demo.tar
* 2)Apply Compression Algorithms on that Archive File: There are multiple compression and decompression algorithms. 1) gzip ◊ It is very fast but less compression power 2) bzip2 ◊ It is a bit slow but more compression power
* To Compress tar file $ gzip demo.tar demo.tar.gz ◊ This file got created 2) To uncompress gz file: $ gzip -d demo.tar.gz OR $ gunzip demo.tar.gz ◊ This command will provide our original tar file
* How to create tar File and compress in a Single Command: 1. By using gzip compression algorithm To create tar and then compress $ tar -cvzf demo.tar \*.txt z option will do compression demo.tar will be created and it is already compressed
* To uncompress and extract tar file $ tar -xvzf demo.tar
* grep stands for globally search a regular expression and print it
* 1) To Search Data in a Single File: $ grep 'durga' subjects.txt
* 2) To Search in Multiple Files: $ grep durga subjects.txt career.txt
* 3) To Search Data by ignoring Case: Bydefault grep command will consider case. If we want to ignore case then we should use use -i option.
* 4) To Display the Number of Occurrences: We have to use -c option. c means count $ grep -c unix \*.txt
* 5) To Display Line Numbers before Results: We have to use -n option.
* ) To Display only File Names in which Pattern exists: We have to use -l option.
* 7) To Print except matched Lines remaining Lines: We have to use -v option. It means inverted.
* 8) To Search for exact word in the File: For this we have to use -w option.
* 9) Display before, after and surrounding lines including Search Results: We have to use -A,-B,-C options -A means after -B means before -C means before and after
* $ grep -C 2 friends demo.tx
* Search Multiple Content in a File: durgasoft@durgasoft:~/Desktop$ grep -i -e "java" -e "unix" demo.txt
* Instead of using -e option, we can use egrep command directly. It is extended grep. It interprets patterns as an extended reqular expression. durgasoft@durgasoft:~/Desktop$ egrep -i "(java|unix)" demo.txt
* grep with -F Option OR fgrep: fgrep ◊ "Fixed String Global Regular Expression Print" It will take a group of fixed strings and search for those in the given file. Strings should be separated by new line.
* Note: fgrep can be used only for Strings and cannot be used for regular expressions.
* Normal grep command cannot understand some regular expression patterns like | symbol. But egrep command can understand any regular expression pattern. Hence egrep is the more powerful than normal grep command.
* In this case, (java|unix) is treated as regular expression. It will search for either java or unix. egrep and fgrep are deprecated and hence it is recommend to use grep -E and grep -F commands. If strings are available in the file, then we can use -f option to specify file name.
* To Print only matched Patterns instead of Total Line: We have to use -o option.
* To Search in the Files recursively inside a Directory: If we want to use grep command for all files present in the specified directory and sub directory recursively then we should use R option.
* Note: If we want to use grep command for directory, compulsory we should use -R option, which means recursive
* Types of Regular Expressions/Patterns: All Regular expression patterns are divided into 3 types. 1) Character Patterns 2) Word Patterns 3) Line Patterns
* 1) Character Patterns: 1) $ grep 'd\*' demo.txt • It display all lines which contains d followed by any number of characters. • ubuntu not providing support for this. 2) $ grep 'c[aeiou]ll' demo.txt It will search for call, cell, cill, coll, cull 3) $ grep 'b..l' demo.txt . means any character. It will search for all 4 letter words where first letter should b and last letter should be l
* 2) Word Patterns: \ ◊ It will always searches for the given word It is exactly same as grep -w word demo.txt \ ◊ It will search for the word ends with xyz
* 3) Line Patterns (Anchors): ^ ◊ Line starts with $ ◊ Line ends with 1) $ grep '^d' demo.txt It will display all lines starts with d 2) $ grep '^the' demo.txt It will display all lines starts with the 3) $ grep '^\' demo.txt It will display all lines starts with the word 'the' 4) $ grep '^[aeiou]' demo.txt It will display all lines starts with vowel. 5) $ grep '^[^aeiou]' demo.txt It will display all lines not starts with vowel. 6) $ grep 't$' demo.txt It will display all lines ends with t 7) $ grep '[aeiou]$' demo.txt It will display all lines ends with vowel 8) $ grep '[0-9]$' demo.txt It will display all lines ends with digit
* ) $ grep '^unix$' demo.txt It will display all lines where total line content should be unix
* 10) $ grep '^....$' demo.txt It will display all lines where line contains exactly 4 characters. 11) $ grep '^\.' demo.txt It will display all lines starts with . 12) $ grep '\$$' demo.txt It will display all lines ends with $ 13) $ grep '^$' demo.txt It will display all blank lines 14) $ grep -v '^$' demo.txt It will display all lines except blank lines Q2) How to Delete Blank Lines Present in the given File? $ grep -v '^$' demo.txt > temp.txt $ mv temp.txt demo.txt Note: $ grep -v '^$' demo.txt > demo.txt The above command won't work grep: input file ‘demo.txt’ is also the output
* Additional Patterns supported by only egrep but not grep: 1. (|) It matches any of the string in the given list $ egrep '(unix|java|oracle)' demo.txt 2. {m} ◊ It matches exact number of preceding character. $ egrep '[6-9][0-9]{9}' demo.txt It will search for 10 digit mobile numbers 3. {m,n} ◊The preceding character should match minimum m times and maximum n times. $ egrep '[0-9]{1,5}' demo.txt It will search for lines which contains minimum 1 digit and maximum 5 digit numbers. 4. {m,} ◊ minimum m number of times but no restriction on maximum number
* We can use cut command to extract data from the file. The file should contain column formatted data, ie tabular data.
* Display Character on specific Position in every Record: $ cut -c 9 emp.dat
* ) Display Range of Characters in every Record: • $cut -c 5-9 emp.dat It will display 5th to 9th characters in every record • $cut -c 5- emp.dat It will display 5th character to last character in every record • $ cut -c -3 emp.dat It will display from 1st character to 3rd character in every record. • $ cut -c 3-5,7-10 emp.dat It will display 3rd to 5th character and 7th to 10th character in every record.
* paste Command: We can use paste command to join two or more files horizontally by using some delimiter. Default delimiter is tab. Syntax: $ paste file1 file2 ...
* $ paste -d '-' subjects.txt fee.txt
* Note: Delimiter should be only one character. If we are providing more than one character, then it will consider only first character. tr Command: tr means translate. This command translates character by character. durgasoft@durgasoft:~/Desktop$ cat > demo.txt While learning unix not required to eat
* User Categories: user/owner ◊ Represented by 'u' group ◊ Represented by 'g' others ◊ Represented by 'o' all ◊ Represented by 'a'
* Permission Types: For files and directories, there are 4 types of permissions. 1) r ◊ Read 2) w ◊ Write 3) x ◊ Execute 4) - ◊ No Permission
* Operations related to permissions: We can perform the following 3 operations. + ◊ Add a particular permission to user|group|other|all - ◊ Remove a particular permission to user|group|other|all = ◊ Assignment a particular permission to user|group|other|all
* chmod Command: chmod means change mode. We can use chmod command to change file or directory permissions. Syntax: $ chmod file\_name/directory\_name
* Eg: For user add execute permission,for group add write permission,for others remove read permission $ chmod u+x,g+w,o-r demo.txt
* Total 9 permissions. First 3 are user permissions, next 3 are group permissions and next 3 are others permissions. user permissions: rw- user can perform both read and write operations but not execute operation group permissions: r-- group members can perform only read operation and cannot perform write and execute operations others permissions: r-- other members can perform only read operation and cannot perform write and execute operations. User Permissions + Group Permissions + Others Permissions order is important
* Read permission + Write Permission + Execute Permission order is important Eg 1: $ chmod u+x demo.txt adding execute permission to the user Eg 2: $ chmod u+w,g+rw,o+r demo.txt adding write permission to the user adding read and write permissions to the group adding read permission to the others Eg 3: $chmod u+x,g-w,o+w demo.txt adding execute permission to the user removing write permission from the group adding write permission to the others Eg 4: $ chmod u=rw,g=rw,o=r demo.txt Now user permissions: rw- group permission: rw- others permission: r—
* Read Permission to the File: If the file not having read permission then we are not allowed to view content of the file. Hence cat, head, tail, more, less commands won't work.
* Write Permission to the File: If the file not having write permission, then we cannot modify the content of the file
* Execute Permission to the File: If the user not has executed permission on any file, then he cannot execute that file as a program
* Read Permission to the Directory: If the user has read permission on any directory, then he can list out the contents of that directory. i.e he can use ls command.
* Write Permission on the Directory: If the user has write permission on any directory, then he is allowed to modify the content of that directory. i.e he can add new files and remove existing files.
* Execute Permission to the Directory: If the user not has executed permission on any directory, then he is not allowed to enter into that directory. i.e he cannot use cd command.
* \*\*\*Note: If the user not having execute permission on any directory, then he cannot perform read and write operations also, because to perform these operations he should enter into that directory which is not possible.
* Linux vs Security: The virus files usually created by others. others are not having execute permission on our directories. Hence others are not allowed to add virus files to our directories. Hackers are not having executed permission on our directories. Hence they cannot read our file data. Because of this, Linux is considered as more secured operating system. Linux follows 2 levels of security. 1st level: login with credentials 2nd level: File and Directory permissions Note: We are using permission types as r,w,x and these are considered as symbolic permissions. But we can also specify permissions by using octal number, such type of permissions are called numeric permissions.
* Numeric Permissions: We can specify permissions by using octal number. Octal means base-8 and allowed digits are 0 to 7. 0 ◊ 000 ⎝ No Permission 1 ◊ 001 ⎝ Execute Permission 2 ◊ 010 ⎝ Write Permission 3 ◊ 011 ⎝ Write and execute Permissions 4 ◊ 100 ⎝ Read Permission 5 ◊ 101 ⎝ Read and execute Permissions 6 ◊ 110 ⎝ Read and write Permission 7 ◊ 111 ⎝ Read, Write and execute Permissions
* Note: 4 ◊ Read Permission 2 ◊ Write Permission 1 ◊ Execute Permission It is more easy to remember 5 ◊ 4+1 ◊ r-x 3 ◊ 2+1 ◊ -wx 6 ◊ 4+2 ◊ rwetc $ chmod 632 demo.txt
* umask Command: umask means user mask. Hiding permissions. Based on umask value,default permissions will be there for files and directories. The default umask value:022 durgasoft@durgasoft:~/Desktop$ umask 0022 First 0 is sticky bit mostly used in admin related activities. We have to consider only last 3 digits as umask value
* Default permissions to the file: 666 - umask value = 666 - 022 = 644 (user ◊ r&w, group ◊ read, others ◊ read)
* Default permissions to the directory= 777 - umask value =777-022 =755 (user ◊ r&w&x, group ◊ r&x, others ◊ r&x)
* chown Command: chown means change owner. Only root user can perform this activity
* chgrp Command: chgrp means change group. Only root user can perform this activity.
* How to Insert and Append Data: A ◊ To Append data at the end of the line I ◊ To Insert data at the beginning of the line a ◊ To append data to the right side of the cursor position (Just after the cursor position) i ◊ To insert data to the left side of the cursor position (Just before the cursor position)
* How to Delete Data: We can delete data either by character wise or by word wise or by line wise. deletion character wise: x ◊ To delete a single character (del key) 3x ◊ To delete 3 characters. Instead of 3, we can pass any number. X ◊ To delete previous character (backspace key) 3X ◊ To delete last 3 previous characters
* Deletion Line wise: dd ◊ To delete current line
* Deletion Words wise: dw ◊ To delete current word.
* Exit Mode Commands: :w ◊ Save File Data :wq ◊ Save and Quit from the Editor :q ◊ Quit Editor :q! ◊ Force Quit. If we perform any changes those will be discarded.