**Linux Documentation**

**Basic Linux CLI command**

* echo Hello 🡪 print Hello in CLI
* cd /tmp 🡪 Change Directory to folder tmp
* cd ~/tmp 🡪 Change Directory to folder $HOME/tmp
* cd / 🡪 go to to root directory, the same as C:\ in windows
* cd .. 🡪 go back one level directory
* cd or cd ~ 🡪 go to home directory
* pwd 🡪 Print Working Directory
* ls 🡪 shows files in current directory
* ls -R 🡪 shows also all files in subdirectories
* ls -l 🡪 shows detail information of the files: type, memory block, owner, user group, size, etc
* ls -a 🡪 shows also all hidden files
* ls -al 🡪 shows detail information of all files: type, memory block, owner, user group, size, etc
* cat 🡪 creating and viewing files: display, copy, combine, create text files
* cat > file1; This is a test file; ctrl+d 🡪 make a text file file1 with given content
* cat file1 🡪 viewing the content of file1
* cat file1 file2 > file3 🡪 concat file1 and file2 to be file3
* rm file1 🡪 delete file
* mv file1 /home/mukhlishga/Files 🡪 move file1 to folder Files
* mv file1 file2 🡪 rename file2 to file2
* mkdir dir1 dir2 dir3 🡪 make a directory
* rmdir dir1 🡪 delete directory
* mv dir1 dir2 🡪 rename directory dir1 to dir2
* man <command> 🡪 open the manual/documentation of a command
* history 🡪 show the commands you have run before
* clear 🡪 clear terminal view
* copy and paste 🡪 ctrl + c, ctrl + shift + v
* sudo 🡪 allows regular user to be a superuser or root
* sudo apt-get install python 🡪 install python
* find . -name ‘my\*’ 🡪 find in current working directory every file starting with ‘my’
* netstat -ntlp 🡪 viewing active port of VM

**Pipes, Grep, Sort CLI commands**

* | 🡪 pipe, to run several commands consecutively
* cat file1 | pg or cat file1 | more 🡪 show file1 contents at the down page
* grep 🡪 scan a document and present the result in a desired format
* cat file1 | grep a 🡪 find the content of file1 containing string ‘a’
* cat file1 | grep -i a 🡪 filter the content of file1 that match (upper and lower) string ‘a’
* cat file1 | grep -v a 🡪 filter the content of file1 that do not match letter ‘a’
* grep -c 🡪 displays only the count of matching lines
* grep -n 🡪 shows the matching lines and its number
* grep -l 🡪 shows just the name of the file with the string
* sort 🡪 sort the content of a text file
* sort file1 🡪 sort the content of file1
* sort -r file1 🡪 reverse sort the content of file1
* sort -n file1 🡪 numerically sorting
* sort -f file1 🡪 case insensitive sorting
* Filtering 🡪 cat + grep + sort etc
* Cat file1 | grep -v a | sort -r
* less, pg, more 🡪 to divide a long file into readable bits

**Regular Expression (RegEx) command**

Commonly used with tr, sed, vi, grep

* . 🡪 replaces any character
* ^ 🡪 matches start of string
* cat file1 | grep ^a 🡪 filter contents start with ‘a’
* $ 🡪 matches end of string
* cat file1 | grep $a 🡪 filter contents end with ‘a’
* \* 🡪 matches up zero or more times the preceding character
* \ 🡪 represent special characters
* () 🡪 groups regular expressions
* ? 🡪 matches up exactly 1 character
* -E {n} 🡪 matches the preceding character appearing n times exactly
* cat file1 | grep -E p\{2} 🡪 filter contents with 2 string ‘p’
* -E {n,m} 🡪 matches the preceding character appearing n times but not more than m
* -E {n, } 🡪 matches the preceding character only when it appears n times or more
* \+ 🡪 matches one or more occurrence of the previous character
* Cat file1 | grep “a\+t” 🡪 filter contents with string ‘at’
* \? 🡪 matches zero or one occurrence of the previous character
* Sequence
* echo {aa,bb,cc,dd} 🡪 print a sequence written between {}
* echo {0..10} 🡪 print a sequence from 0 to 10
* echo {a..z} 🡪 print a sequence from a to z
* echo a{0..9}b 🡪 print a sequence from a0b to a9b

**Process management in Linux**

process = an instance of a program. Every command start a new process

foreground process 🡪 run on screes, need user input 🡪 start program, press ctrl + z, type fg

background process 🡪 run in background, not need user input 🡪 start program, press ctrl + z, type bg

* top 🡪 display all the running processes:
* PID 🡪 process ID, unique
* USER 🡪 process’ owner
* PR 🡪 priority (-20 to 20)
* NI 🡪 niceness
* VIRT 🡪 virtual memory used in KB
* RES 🡪 physical memory used in KB
* SHR 🡪 shared memory used in KB
* S 🡪 status: D (uninterruptible), R (running), S (sleeping), T (Traced or Stopped), Z (zombie)
* COMMAND 🡪 command’s name
* ps 🡪 Process Status, like task manager in windows
* ps ux 🡪 display all information of process and programs running under user
* pidof processName 🡪 find the PID of a process
* ps PID 🡪 display status of a single process
* kill 🡪 terminate running process
* pidof processName 🡪 find the PID of a process
* kill PID 🡪 terminate a single process
* Niceness 🡪 priority of process, -20 (highest priority) to 19 (lowest priority), default is 0
* nice -n niceValue processName 🡪 launch a process while giving it priority
* renice niceValue -p PID 🡪 give priority to a already running process
* df -h 🡪 report the free disk space
* free -m or free -g 🡪 show free and used memory (RAM) on Linux in MB or GB
* lsof -i tcp:8080 🡪 get the PID of process running on tcp:8080

**VI editor in Linux**

* vi file1 🡪 create or edit a text file named file1
* vi operation:
* command mode 🡪 VI editor opens, cursor move, cut copy paste, save
* insert mode 🡪 press ‘i’ in command mode, press ‘esc’ to save and return to command mode
* vi editing command:
* i 🡪 insert at cursor (goes to insert mode)
* ESC 🡪 terminate insert mode
* shift + zz 🡪 save and quit
* :w 🡪 save but keep the file open
* :q 🡪 quit without saving
* :wq 🡪 save and quit

**Shell scripting in Linux**

Prime components of operating system:

Kernel 🡪 nucleus of computer, communication between software and hardware

Shell 🡪 the interface, takes input from user in form of command, process it, and make output

1. Bourne shell 🡪 dollar sign ($) prompt

* POSIX shell 🡪 sh
* Korn shell 🡪 sh
* Bourne Again SHell 🡪 bash

1. C Shell 🡪 percent sign (%) prompt

* C shell 🡪 csh
* Tops C shell 🡪 tcsh
* Shell scripting 🡪 writing a series of commands for shell to execute, the lengthy and repetitive commands can be stored and executed anytime
* create a file using a text editor like vi
* name the script file with extension .sh
* start the script with #! /bin/sh 🡪 shebang operator that directs the script to interpretation location
* write some code, add # for comment
* save the script file as filename.sh
* to execute 🡪 bash filename.sh
* shell variable 🡪 can be read by the shell only. 'read’ command used to capture input
* e.g.:

#! /bin/sh

echo “what is your name?”

read name

echo “hello $name!”

* can also use conditional statements, loops, functions

**Perl programming**

Perl 🡪 language specially designed for text editing: linux system administration, network programming,

* Perl is more opted than shell scripting because:
* Does not cause portability issues
* Error handling is very easy
* Writing long and complex (OOP etc) is easier
* More reusable library (CPAN)
* More secure
* Perl scripting
* start the script with🡪 #!/usr/bin/perl 🡪 location to perl interpreter
* make variable 🡪 $variable = “value”; 🡪 $name = “Ronald”;
* output 🡪 print(“something”); 🡪 print(“thanks”);
* input 🡪 $variable = <STDIN>; 🡪 $username = <STDIN>;
* comment start with #, end every statement with ;, save file with .pl extension
* to execute 🡪 perl filename.pl
* e.g.:

#! /usr/bin/perl

print(“what is your name?”);

$name = <STDIN>;

print(“hello $name!”);

**Environment variables**

* env 🡪 display all environment variables
* echo $VARIABLE 🡪 change VARIABLE with the following:
* PATH 🡪 contains colon (:)-separated list of directories in which the system looks for executable file. Windows OS has this env variable
* USER 🡪 username
* HOME 🡪 default path to user’s home directory
* EDITOR 🡪 path to the program which edits the content of files
* UID 🡪 user’s unique ID
* TERM 🡪 default terminal emulator
* SHELL 🡪 shell being used by the user
* ENV 🡪 displays all the environment variables
* NEWVARIABLE=value123 🡪 assign NEWVARIABLE with ‘value123’
* unset NEWVARIABLE 🡪 delete NEWVARIABLE variable

**Print CLI command**

* pr -3 file1 🡪 print file1 (text file) with 3 columns
* pr -h “header” file1 🡪 print file1 (text file) with a header containing “header”
* pr -t file1 🡪 does not print the header and top/bottom margins
* pr -d file1 🡪 double spaces the output file
* pr -n file1 🡪 denotes all line with numbers
* lpr n file1 🡪 print file1 for n copies

**Redirection in Linux**

All file has File Descriptor (FD) 🡪 FD0 (stdin), FD1 (stdout), FD2 (stderr)

You can redirect error using its corresponding File Descriptor 2 (FD2), for example when you search a file and it is not exist, store the error message to a file

* > 🡪 stdout, outputting certain commands, e.g. echo hello > file1 🡪 file1 contains “hello”
* >> 🡪 appending instead of overwriting, e.g. echo world >> file1 🡪 file1 contains “hello world”
* < 🡪 stdin
* >& 🡪 redirect output of one file to another.
* find . -name ‘my\*’ 2> error.log 🡪 find file, if error then store error in error.log file
* ls Documents ABD > dirlist 2>&1 🡪 find file ABC in directory Documents, fail, error says that the list of Documents are file1, file2, then store this message t dirlist file

**Add and delete users and groups**

* add new user
* sudo adduser user1 🡪 adding new standard user named user1
* delete user
* sudo passwd -l user1 🡪 disabling user1 password
* sudo userdel -r user1 🡪 deleting user1 account
* add user to group
* groupmode + tab key twice 🡪 viewing existing groups
* sudo usermod -a -G group1 user1 🡪 adding user1 to group1
* cat /etc/group 🡪 check if user in group
* remove user from group
* sudo deluser user1 group1 🡪 remove user1 from group1
* using GUI apps to manage user and group
* sudo apt-get install gnome-system-tools
* finger 🡪 information of the users on a Linux machine
* finger user1 🡪 information of user1 account

**Linux file permission**

* ownership 🡪 u (user), g (group), o (other), a (all)
* permission 🡪 r (read), w (write), x (execute), - (no permission)
* symbolic mode 🡪 + (add permission), - (remove pemission), = (set and override permission)
* chmod a=rwx file1 🡪 all user can read, write, execute file1
* ls -l 🡪 shows detail information of the files, first column 🡪 type of permission
* chown user1 file1 🡪 user1 now owns file1
* chown user1:group1 file1 🡪 user1 and group1 now owns file1
* groups 🡪 to list groups. 2 groups cannot own the same file

**Communication in Linux**

* ping 🡪 check if your connection to the server is healthy
* ping 172.16.170.1 🡪 ping to server with ip address of 172.16.170.1
* ping www.google.com 🡪 ping to google
* FTP 🡪 File Transfer Protocol
* ftp <ip-address or hostname> 🡪 log into remote computer
* dir 🡪 display files in current directory of remote computer
* cd ‘dirname’ 🡪 change directory on remote computer
* put file1 🡪 upload file1 from local to remote computer
* get file1 🡪 download file1 from remote to local computer
* quit 🡪 logout
* telnet 🡪 connect to a remote Linux computer and run program remotely 🡪 obsolete technology
* telnet <ip-address or hostname> 🡪 log into remote computer
* SSH 🡪 Secure Shell, securely connect to a remote Linux computer
* ssh username@ip-address or hostname 🡪 log into remote computer

Convention

* # – requires given linux commands to be executed with root privileges either directly as a root user or by use of sudo command
* $ – requires given linux commands to be executed as a regular non-privileged user

In window, file is stored in local disk C, D, E

In linux, data is starting with the root directory. Root 🡪 etc, bin, usr, tmp, dev

In linux 🡪 file, directory, devices all are file

Linux user: regular user, root user, service account

**Creating a Ubuntu Linux VM**

1. Install VirtualBox
2. Download ubuntu iso file
3. Follow this tutorial:

https://www.youtube.com/watch?v=x5MhydijWmc&ab\_channel=ProgrammingKnowledge

Regular user:



Root user:

