**Red Hat Certification System Administrator**

# Linux Architecture

The kernel is linked directly to the hardware components. The shell is a higher level than the kernel and we have the highest level that is the closest to the user which is the user interface.

# Check status of services

**systemctl** **status** sshd : This command will check the status of the service sshd which is the service responsible for remote connections on servers (OpenSSH)

# Check ip address

**ip addr** : To see the interfaces and ip addresses.

# Relative path vs Absolute path

Absolute path is when we start from the root directory and pass to another directory (/home/elie)

♣ Commands are case sensitive

# Open a file and write in it

**cat >** firstFile

Here we can write some text but if there were some info in the file before it will be overridden.

**cat >>** firstFile

Here we will not override we will just add.

**cat -n >>** firstFile

Will add the number for each line

# Traverse between directories

**cd -** : Go to the last directory.

**cd ~** : Go to home directory.

# Work with directories

**mkdir -p** 1/2/3/4 : Here we created 4 direcotires nested in each other so the the directory 1 is the first parent.

**mkdir** {2008..2021}-{01..12}

**rmdir 6/ :** Removes the directory 6 but in condition that the directory is empty otherwise we should add the parameter **-f**

**rmdir -rf** 4/ : ‘r’ means recursif so here it will delete nested directories. We can also add the **-i** that will ask for confirmation before deleting.

# Check the history of commands

**history** : Show all executed commands and each command will be listed on a line with the number of the command, let’s say I want to execute the command of the line 71 I can do : **!71**

**!!** : Will execute the last command

**history -c** : Will delete the history

# Check info about machine and os

**uname -a**

# File ownership

Owner : Group : World : Path

4 read (r)

2 write (w)

1 execute (x)

7 = 4+2+1

6 = 4+2

**ls -l** file ou **ll** file:

- rw- rw- r--

The - determines a file or directory or link.

**chmod** 554 file : Will give the following access rules for the file.

**chmod +x** : Add execute permission to owner and group of file.

**chmod u+rw** : Add r/w to user.

**chmod -R+x** : Add execute permissions to files and directories permissions recursively.

**chmod g-rw** : Removes read and write for group

**chmod ugo+rwx** : Adds r/w/x for user,group and others.

**chmod -R 770** Downloads/ : Changes recursively the permissions for this directory.

**chown** root file : Change the owner group of file so root is owner of group

**chown** root : root file : Changes the owner and group of file.

**chgrp** root : file : Change the group to root for the file.

# Work with files

**touch** file-‘date +%d-%M-%Y’

Result : file-4-12-2021

**touch** file{1..100}.txt

Here we will create files going from 1 to 100

# Copy some files

**cp** .txt Downloads

Will copy all text files to the Downloads directory.

**cp -pv** file Downloads/

The ‘-p’ is used to preserve parameters so for example the modification date of the copied file will be same as before copying. The ‘-v’ means verbose which will show us a sentence explaining what happened.

**cp -R** Desktop/ Downloads/

The ‘-R’ is used to copy a directory.

**scp** backup.tar.gz root@192.168.2.140:/opt/ : Will copy file to opt directory on the remote server.

# Move some files

**mv** file othernamefile : Renames the file.

**mv** file Downloads/ : Moves to Downloads.

# List some files

**ls -m** /home/elie : ‘-m’ is used to list files separated with a comma. Usually used when exporting.

**ls -r** : list subdirectories also.

**less** : Display file content 1 page at a time

# Create a link

**ln -s** Kenlm/LICENSE : Create link to that file with same name

# Compare two files

**diff** file1 file2 : Verify if two files are equal

# Output some data

**head** : By default, prints first 10 lines of file

**head -n** 15 : First 15 lines

**tail** : last 10 lines

**tail -n** 20 /var/log/messages **>** /home/aravi/logfile

# Arithmetic expression

**echo $**((7+3)) : Will output 10

# Brace expansion

**echo** abc{elie,joe}xyz : Will output 2 things : abceliexyz and abcjoexyz

**echo** {090..100} : 090 091 092 … 100

**echo** a{A{1,2},B{3,4}}b : aA1b aA2b aB3b aB4b

# Work with echo

**echo $**{USER} : Will output the value of the variable USER

**echo \**$USER : Outputs \USER

**echo** my name is \\$USER : Will output my name is elie because the \\ will cancel the one the other.

# Work with partitions

**df -h** : Will show us partitions and directories with some details.

**du -sh** /etc/\* : Shows directories size inside of etc directory.

# Work with users

**who** : Used to see the users currently logged.

# Work with processes

**ps -aux** : Show all running processes.

**ps -U** elie : Show the processes running by the user elie.

# Get help for a certain command

**ls --help**

**man ls**

# Search for files

**locate** or **find** : Locate is faster but need database update. Find is used when we know where the file is located.

**updatedb** : Will do a database update

**find** /home/ **-i -name** “file1” : The ‘-i’ will ignore the case sensitive.

**locate -i** fil : If you know a fragment of file

**locate -ib** fil : Without the ‘-b’ it will take the path as a search parameter so it will output everything in it. The ‘-b’ will force to use the last thing we wrotes in the command.

# Archieve and compress

Archieving is different than compressing. Archieving means putting files into one file without reducing size so without compressing.

We have 3 types of zip methods :

- gzip

- bzip

- zip

**tar -cvf** archieve.tar varlog.tar.gz anothervar.rpm : The “-c” means create, the “-v” means archeive and the “-f” means specify files. So here we archieve files into one file without compressing.

**tar -cvzf** archieve.tar varlog.tar.gz anothervar.rpm : The “-z” will zip the resulting file.

**tar -uvf** archieve.tar \*.txt : We can update the contents of the archieve file so for example if we made changes in the folder and want to recompress it we can just update. Here we are saying to check all text files that has been changed to update the zip folder in the end.

**tar -tf** archieve.tar : The “-t” will list the files that are present in the archieve folder.

**tar -xvf**  archeive.tar : Extracts files from archieve.

**zip** -9 **-r** test1.zip /home/aravi : Here we used zip so it is another method and “-9” let us do high compression and here we are compressing all subfiles into one archieve which is tes1.zip

**zip -d** test1.zip home/aravi/ioen/testing.txt : The “-d” is used to delete specific file from the archieve.

**zip -fr** test1.zip \* : Here the test1.zip will be updated.

**unzip** test1.zip : To unzip file

# Redirection of errors

Standard input : 0

Standard output : 1

Standard error : 2

**ls** /root 2**>** /temp/errors : Will insert the error line into errors file and the ‘2’ refers to the standard error value.

**find** /etc **-name** passwd **>** /tmp/output 2 **>** /tmp/error : Will redirect the output file to /tmp/output and if there is an error, the /tmp/error will be filled with the error line.

**ls** /home/aravi **>>** /tmp/errors 2 **>** &1 : The **>>** will let insert multiple things instantly so here we insert the output and the error if there is one.

# Pass output as input for a command

**grep** aravi **<** /tmp/output : Will give the output of /tmp/output to the command as an input

# Grep command

**grep -e** “First” **-e** “last” demofile : So the ‘-e’ must be used if we want to search multiple strings

**grep -v** “First” demofile : The ‘-v’ is used to unmatch the word so we will output everything except the word “First”.

**grep** “First” **-A** 2 file : Will output line containing the word “First” and 2 lines after.

**grep** “First” **-B** 2 file : Will output line containing the word “First” and 2 lines before.

**grep** “All” **-C** 1 file : Will output 1 line before and 1 line after.

**grep -ril** /home/aravi/ : The “-l” is used when searching files or directories.

**grep -l** demo\* : Will output the files having “demo” included in their name.

**grep -c** first demofile : Will output the number of times the word “first” is found in demofile.

**grep** “demo**$**” demofie : Will output the line that ends with the word “demo”.

**cat** /var/log/messages **| grep** “^Sep 26 03:26:41” : Will search for lines beginning (^) with date.

# Vi&Vim editor

Vi is the best editor to use in Linux. It is used to modify config files.

The difference between Vi and Vim is that Vim is an imoproved version of Vi where output is colored.

# Manual page

**man -s** 5 passwd : Will give us passwd(5) which is the 5th sub category.

**man -k** printf : Shows us how many related command are there. So all commands related to printing.

**man -s** 5 **-k** passwd : Show the command linked to passwd in the 5th sub category.

♣ In the manual page I can enter / and search.

**pinfo** ls : Similar to man command but different the way and content.

**info** ls : Exactly similar to man command.

# whereis and whatis command

**whereis** python : Show us where it is located

**whatis** ls : Brief description of command

# SSH Servers

Here we are going to explore OpenSSH server&client configuration so how to securely log in to the server and how the client is authentified. OpenSSH use port 22 by default for communicating between client and server. Public and private keys are used for cryptographics purposes.

- Firstly we must install the OpenSSH package :

**yum install** openssh

♣ Difference between yum and rpm : Yum is a package manager and rpms are the actual packages. With yum you can add or remove software. The software itself comes within a rpm.

**rpm -qa | grep** openssh : Will show us the installed version of openssh, so here if we are on the server side we will see the installed package.

- Now we have to configure the config files and SSH keys on server side and authenticate the client.

**systemctl status** sshd : Will show us the status of the service openssh.

**systemctl enable** sshd : To enable the service openssh if it is not enabled.

- Now we must verify the config files :

**vi** /etc/ssh/sshd-config : Now when we do access of the file we can add some code

a) Add the protocol type responsible for a secured environnement :

Protocol 2

UsePAM yes

AllowGroups sshusers (This group only will have access to the ssh server)

- Now we will create the group

**groupadd** sshusers

**usermod -Ag** sshusers aravi : Add user aravi to group sshsuers

**id** aravi : To verify that aravi is part of the group sshusers.

- Now we will try to connect to the server

**ssh** aravi@192.168.2.140 : Here he will ask for passwd of aravi and we will be able to connect to ssh server.

♣ I can also use Putty and choose the connection type as SSH and he will also ask for user’s password.

# SCP (Secured copy)

In this section, we will learn how to securely copy files from a source to another.

- Source IP Server : 192.168.2.140

- Destination IP Server : 192.168.2.42

**scp** file1 root@192.168.2.42:/root/ : The ip address and location in orange is the ip address of the destination server and the location where we want to store the file.

We can add multiple files simulatenously

**scp -r** directoryName root@192.168.2.42:/root/ : Here we used the “-r” to copy entire directory.

**scp -C** filename root@192.168.2.42:/root/ : The “-C” is used to compress and send data.

In fact, in this situation the timestamp will be the actual one when we make the copy that’s why we can use the parameter “-p” to preserve parameters.

**scp -vp** filenameroot@192.168.2.42:/root/ : To preserve parameters

**scp -l** 500 filename root@192.168.2.42:/root/ : To limit the bandwidth allocated for the copy.

# Linux Processes

**sleep** 300 **&**

output : [1] 2798 : This is the process ID.

**ps -aux | grep** 2798

In fact, each process has a specific state :

A) Running : Running or ready to run

B) Waiting : Process is waiting for an event or for a resource (Like waiting to be allocated a certain part of CPU or Ram.

C) Stopped : Received a stop signal

D) Orphaned : Process exists while children still running, thoses childrens are orphanes.

E) Zombie : It is a halted process. Still has a task\_struct data structure in task vector. .Here the process is dead but still found in Process table. To find these processes:

**ps -aux | grep** Z

**pstree** : Shows us the process tree containing father and children processes.

**ps -ef** : Show us not only processes like **ps -a** but also sleeping processes and all processes.

**ps -aux --sort = -pcpu,+pmem**: To sort process with utilization rate.

**ps -e -o** : Filter with , so we will see pid,uname,pcpu,pmem…

**jobs** : [1] Running

**fg %**1 : Here the ‘fg’ means foreground so it will kick the process having id ‘1’.

**kill -l** : Will show us all parameters that I can give to the kill command and each parameter is identified with a specific ID or number.

**kill** -9 3298 : Here we will kill this process. (-9) means kill process.

**pgrep -u** root ssh: See all executed commands with root processes about ssh.

Change process priroty : A normal user can only decrease a process priority but not increase it. The ‘-20’ is the highest priority value and the ‘20’ is the lowest.

**nice** -10 **sleep** 500 **&** : Will give a value of ‘-10’ as priority for this process.

# Disk partitioning

**sudo fdisk -l** : To see the hardrives or partitions.

♣ You can make only 4 partitions per hardrive (for Windows or Linux). In fact, we use the 4th partition for example as an extended one where inside of it we can use multiple partitions (locial partitions)

**sudo -s** : Go to root

**fdisk** /dev/sdb : Then enter ‘m’ for help then we will see multiple parameters to choose from and we will type ‘n’ to add a new partition. Then we will choose ‘p’ for primary then we will choose a partition number (1🡪4). Then we will choose the size of the 1st partition, we will put +5G then we will enter ‘wq’ to write and quit.

**partprobe** /dev/sdb : To update the partition table to this hardrive.

Now we will create or directory and add it (mount it) to a specific partition.

**mkdir** /part1

**mount** /dev/sdb1 /part1/ : It will not work and return an error, we need to add a file system to the partition.

**mkfs**.ext4 /dev/sdb1

**mount** /dev/sdb1 /part1/

♣ Difference between ‘df -h’ and ‘fdisk’ :

df -h : Displays all the mounted partitions.

fdisk -l : Displays all partitions that exist on your disk. Apparently, there are some partitions which exist but which are not mounted

**df -h** : To see all mounted partitions. Now here we will notice that the mount didn’t actually work, we should add something in /etc/fstab

**vi** /etc/fstab

Then we will add the following line :

/dev/sdb1 /part1 ext4 defaults 0 0

**mount -a**

**df -h** : Now we will see that part1 is mounted to sdb1

# Delete partitions

**vi** /etc/fstab

Then we will delete the following line :

/dev/sdb1 /part1 ext4 defaults 0 0

**unmount** /part1

**fdisk** /dev/sdb : Then ‘p’ to print partitions then ‘d’ to delete and choose the partition number, we will choose ‘1’ which refers to sdb1 then ‘wq’ to save and quit.

**partprobe** /dev/sdb

# Create LVM

We used standard partitions in the previous section. One of the disadvantages is that if we want to increase a partition size we need to stop the users using the partitions so the partition will be shut down until we increase it and finish and remount it again. LVM or Logical Volume Manager combines multiple hardrives so if we want to write a file instead of writing in on a single partition which will take specific time, with LVM the time will be split on the number of hardrives so the file will be written in equal manner on the hardrives and this volume group that combines multiple hardrives can be splitted on as many slices that we want. Each slice is called logical volume. So we will have logicalvolume1, logicalvolume2, etc..

In our example, we create 2 partitions. One of them is sdb1 and the other is sdc1. Sdb1 is the partition in sdb hdd and sdc1 is the partition in sdc hdd. In fact, these partitions are standard partitions. We need to convert them to LVM partitions. To do so :

**fdisk** /dev/sdb : Then ‘m’ then ‘t’ then ‘L’ to list all codes then ‘8e’ which is the code of Linux LVM. Now we converted the partition from standard to Linux LVM. Now we will pass to sdc.

**fdisk** /dev/sdc : Then ‘m’ then ‘t’ then ‘L’ to list all codes then ‘8e’ which is the code of Linux LVM. Now we converted the partition from standard to Linux LVM. Now we will pass to sdc.

**partprobe** /dev/sdb

**partprobe** /dev/sdc

Now we have to create the physical volumes inside of each partition

**pvcreate** /dev/sdb1

**pvcreate** /dev/sdc1

**pvs** : To see the created pv’s.

Now we will create the volume group

**vgcreate** VG0/dev/sdb1 /dev/sdc1

**vgs** : To see created vg’s.

**vgdisplay** VG0 : Will output further details on VG0.

**lvcreate -n** lv0 **-L** 4G VG0 : Here we created the logical volume containing the volume group. Now we will create another logical volume.

**lvcreate -n** lv1 **-L** 2G VG0

Now as we created pv’s then vg then lv’s we will create file systems and mount some things to them.

**mkfs**.ext4 /dev/VG0/lv0

**mkfs**.xfs /dev/VG0/lv1

**mkdir** /ext4part

**mkdir** /xfspart

**vi** /etc/fstab

We need to add the following lines :

/dev/VG0/lv0 /ext4part ext4 defaults 1 2

/dev/VG0/lv1 /xfspart xfs defaults 1 2

**mount -a**

**df -h**: To check

# Extend LVM

Here we will show how to extend filesystem without interrupting applications.

FREE SPACE

**LV0**  **Extend LV0** **VG0**

In fact, to be able to extend without interrupting user’s applications we need to resize the file system to this extended value.

**lvextend -L** +1G /dev/VG0/lv0

**resize2fs** /dev/VG0/lv0

♣ For xfs, the below command will change

**lvextend -L** +1G /dev/VG0/lv1

**xfs\_growfs** /dev/VG0/lv1

# Access Linux

● What is swap file system and how it works?

In fact, swap file is used to optimize the load on the physical memory. if we have some apps that are opened we may fill our memory capacities. In order to optimize that we will remove the apps that are inactive for a certain time and put them on the hardisk so we will always have the info but in another location.

Swap memory = Ram X2

Now we will create the partition as swap.

**fdisk** /dev/sdc : Then ‘m’ then ‘n’ then ‘p’ then ‘+2G’ then ‘t’ then ‘1’ then ’82’ which will convert the file system of sdc1 partition from Linux to Linux swap/Solaris. Then ‘wq’ to save.

**partprobe** /dev/sdc

**mkswap** /dev/sdc1

**vi** /etc/fstab

Then add the following :

/dev/sdc1 swap swap defaults 0 0

**swapon -a**

**swapoff -a** : To turn off swap

**swapon -s** : To see the swap partition.

# Create users