---------------------------------------------------------------------------------------------------------------------------------

LINUX

-----------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------

17-01-2022 Class - 1

-------------------------------------------------------------------------------------------------------------------------------------

**Step 1:** Create AWS Account

**Step 2:** Create Redhat Linux Server (t2.micro)

Search for EC2 in the AWS console

Goto Instances on the left and select Launch Instances

Select Redhat Linux 8 flavor of Linux

Select t2.micro in Instance Type

Configure Security Groups

Type - All Traffic

Source - Anywhere

Key Pair

Select create a new key Pair, give a name and download the pem file

-----------------------------------------------------------------------------------------------------------------------------------

**Step 3:** Download and install Putty

**Step 4:** Open Puttygen and ceate a ppk file (converting pem file to ppk)

Puttygen --> Load --> Select the .pem file (All files) --> Save Private Key (ppk)

**Step 5:** Connect to the Linux Server

Open Putty

Copy the public ipv4 address from AWS and paste it in Host Name

Under Connection --> SSH --> Auth --> Select the ppk file that was generated using Puttygen

Login As: ec2-user (Default Username for Redhat Machine)

------------------------------------------------------------------------------------------------------------------------------------

**Linux Commands:**

-------------------------------------------------------------------------------------------------------------------------------------

mkdir <directory\_name> --> To create a directory\_name

touch <file\_name> --> To create a file

--------------------------------------------------------------------------------------------------------------------------------------

ls --> To list the files and directories

ls -l --> To list the files and directories in long format

ls -lt --> To list the files in long format sorted with time

ls -lrt --> To list the files in long format sorted with time in reverse order

----------------------------------------------------------------------------------------------------

cd <directory\_name> --> To change a directory

cd .. --> To go back to a previous directory

cd --> To goto home directory (/home/ec2-user)

--------------------------------------------------------------------------------------------------------------------------------------

pwd --> To check the present working directory

--------------------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------------------

18-01-2022 Class - 2

--------------------------------------------------------------------------------------------------------------------------------------

VI - Text Editor

vi <file\_name> --> To open the file for edit

Note: By default VI editor opens in command mode

esc --> To go back to command mode

esc + i --> To enter Insert mode in VI

esc + :w --> To save the file

esc + :wq --> to save the file and quit

esc + :wq! --> To save the file and quit

esc + :q! --> To quit without saving the file

w --> write

q --> quit

! --> Forcefully

--------------------------------------------------------------------------------------------------------------------------------------

esc + :set nu --> TO set the numbers for the file

esc + :set nonu --> To remove the numbers

--------------------------------------------------------------------------------------------------------------------------------------

cat <file\_name> --> To check the contents of the file

--------------------------------------------------------------------------------------------------------------------------------------

Assignment: How to display the contents of the file in the reverse order

tac <filename>

--------------------------------------------------------------------------------------------------------------------------------------

Find and Replace strings

esc+ :%s/<old\_string>/<new\_string>/g

% --> All lines

s --> Substitute

g --> globally (All occurences of the pattern)

ig --> Case insensitive replace

esc + :1s/<old\_string>/<new\_string>/g --> To replace only in the first line

esc + :1,4s/<old\_string>/<new\_string>/g --> To replace from lines 1 to 4

esc + :4,$s/<old\_string>/<new\_string>/g --> To replace from line 4 to end of the file

--------------------------------------------------------------------------------------------------------------------------------------

esc + dd --> To delete a line in VI

esc + /pattern

--------------------------------------------------------------------------------------------------------------------------------------

esc + :undo

esc + :redo

--------------------------------------------------------------------------------------------------------------------------------------

Remove the files

rm <file\_name> --> To delete a file

rm -rf <directory\_name> --> To delete a directory

rm -rf \* --> Remove all the file and directories in that particular folder

--------------------------------------------------------------------------------------------------------------------------------------

Copy the files

cp <file\_name\_1> <file\_name\_2> --> to copy a file

cp <file\_name\_1> <dir>/<file\_name\_2> --> To copy a file to a directory

cp -R <dir\_1> <dir\_2> --> To copy a directory

--------------------------------------------------------------------------------------------------------------------------------------

Move the files or rename the files

mv <file\_name\_1> <file\_name\_2> --> to move a file

mv <file\_name\_1> <dir>/<file\_name\_2> --> To move a file to a directory

mv <dir\_1> <dir\_2> --> To move a directory

--------------------------------------------------------------------------------------------------------------------------------------

Echo --> Print

echo "hello" --> To print hello on the terminal

echo -e "hello \n how are you" --> To print with new line using \n

--------------------------------------------------------------------------------------------------------------------------------------

Redirect(>) and Append(>>)

echo "hello" > <file\_name> --> Redirect the output of echo command to the file

echo "hello" >> <file\_name> --> Append the new info to the existing file

--------------------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------------------

20-01-2022 Class - 3

--------------------------------------------------------------------------------------------------------------------------------------

Word Count

wc <file\_name> --> The total number of lines, words and character present in the file

wc -l <file\_name> --> To check only the number of lines

wc -w <file\_name> --> To check only the number of words

wc -c <file\_name> --> To check only the number of characters

----------------------------------------------------------------------------------------------------

File size and Disk Size

df -h --> To check the disk size of the system

free -h --> To check the system memory (RAM)

du -sh <file\_name> --> To check the file size

du -sh \* --> To check the size of all files and directories in the pwd

----------------------------------------------------------------------------------------------------

Grep - Used to search for strings inside a file

grep "<string>" <file\_name> --> To display all the lines with the string

grep -i "<string>" <file\_name> --> To display all the lines with the string but case insensitive search

grep -e "<string1>" -e "<string2>" <file\_name> --> To search for multiple patterns

grep -w "<string>" <file\_name> --> To search for a whole word

grep -l "<string>" \* --> To check for the pattern in all the files and prints the filenames

grep -l -R "<string>" --> To check for the pattern in all the files recursively and prints the filenames

grep "^<string>" <file\_name> --> TO print all the lines that start with the pattern

grep "<string>$" <file\_name> --> TO print all the lines that end with the pattern

grep -c "<string>" <file\_name> --> TO count the number of lines the pattern is present

grep -v "<string>" <file\_name> --> To display all the that does not have the pattern

Assignment: Grep Flag to check all the files that does not have the pattern

grep -riL "foo"

-L, --files-without-match

each file processed.

-R, -r, --recursive

Recursively search subdirectories listed.

-i, --ignore-case

Perform case insensitive matching.

----------------------------------------------------------------------------------------------------

SUDO --> Super User Does

sudo gives permissions for users to execute commands as a root user

sudo su --> To goto root user

sudo <command> --> To execute a command when you get permission denied

To give an user the sudo permission, User needs to be added to the sudoers file

File location: /etc/sudoers

<user\_name> ALL=(ALL) NOPASSWD: ALL

----------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------

21-01-2022 Class - 4

----------------------------------------------------------------------------------------------------

Add Users

sudo useradd <user\_name> --> To add a user to the server

sudo passwd <user\_name> --> To set a password for the user

sudo userdel <user\_name> --> to delete a user from the server

To check all the users in the server

getent passwd

cat /etc/passwd

----------------------------------------------------------------------------------------------------

Add Groups

sudo groupadd <group\_name> --> To add a group to the server

sudo groupdel <group\_name> --> to delete a group from the server

sudo usermod -aG <group\_name> <user\_name> --> To add an user to the group

To check all the groups in the server

getent group

cat /etc/group

----------------------------------------------------------------------------------------------------

Permissions in Linux

r - Read Permission

w - Write Permission

x - Execute Permission

-rw-rw-r--

rw- --> Owner of the file

rw- --> Group to which the file belongs

r-- --> Others

r - 2^2 --> 4

w - 2^1 --> 2

x - 2^0 --> 1

- --> 0

rwx rwx rwx

chmod 777 <file\_name> --> To give rwx for owner, group and others

chmod 766 <file\_name> --> To give rwx for owner, rw for group and other

chmod u+x <file\_name> --> To give the owner execute permission

chmod o-x <file\_name> --> TO remove execute permission for others

u --> owner

g --> group

o --> others

----------------------------------------------------------------------------------------------------

Change ownership of files and directories

chown <user\_name> <file\_name> --> To change the owner of the file

chgrp <group\_name> <file\_name> --> To change the group to which the file belongs

chown <user\_name>:<group\_name> <file\_name> --> TO change the owner and the group

----------------------------------------------------------------------------------------------------

Default Permisssion in Linux:

Files --> 666

Directories --> 777

umask 002 --> To change the default permissions of the files and directories to 664 and 775

umask 022 --> To change the default permissions of the files and directories to 664 and 775

/etc/profile --> To change the umask value permanently

----------------------------------------------------------------------------------------------------

head --> To print required starting number of lines

head -n <file\_name> --> To display the starting n lines

head -3 <file\_name> --> To display the starting 3 lines

head <file\_name> --> To display the starting 10 lines

tail --> To print the required lines from the end

tail -n <file\_name> --> To display the last n lines

tail -3 <file\_name> --> To display the last 3 lines

tail <file\_name> --> To display the last 10 lines

----------------------------------------------------------------------------------------------------

Pipe [|] --> To pass the output of one command to the next command

Example:

head -4 <file\_name> | tail -1 --> To display only the 4th line

----------------------------------------------------------------------------------------------------

Sed --> Stream Editor used to find and replace words

sed 's/<old\_string>/<new\_string>/g' <file\_name> --> To replace old string with the new string and

display it on the terminal (It won't make changes to file)

sed -i 's/<old\_string>/<new\_string>/g' <file\_name> --> It will also make changes to the original file

sed -i '1s/<old\_string>/<new\_string>/g' <file\_name> --> It will make changes to the first line only

sed -i '2,5s/<old\_string>/<new\_string>/g' <file\_name> --> It will make changes from 2 to 5th line

sed -i '4,$s/<old\_string>/<new\_string>/g' <file\_name> --> It will make changes from 4 to end of file

sed -i '4d' <file\_name> --> Deletes the 4th line

sed -i '3,5d' <file\_name> --> Deletes the lines from 3 to 5

sed -i '4,$d' <file\_name> --> Deletes the lines from 4th to end of the file

sed -n '2p' <file\_name> --> To print the 2nd line only

sed -n '2,4p' <file\_name> --> To print the lines from 2 to 4

sed -n '2,$p' <file\_name> --> To print the lines from 2nd to end of the file

----------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------

22-01-2022 Class - 6

----------------------------------------------------------------------------------------------------

Cut - Used to cut a file column wise

cut -d " " -f1 <file\_name> --> To display only the 1st column

cut -d " " -f1,3 <file\_name> --> To display the 1st and 3rd column

cut -d " " -f2-5 <file\_name> --> To display from 2 to 5th columns

d --> Delimiter

----------------------------------------------------------------------------------------------------

awk '{print}' <file\_name> --> TO display the contents of the file

awk -F " " '{print$2}' <file\_name> --> To display only the 2nd column

awk -F " " '{print$2,$3}' <file\_name> --> To display the 2nd and 3rd column

awk -F " " '{print$NF}' <file\_name> --> To display the last column

awk -F " " '{print$(NF-1)}' <file\_name> --> To display only the 2nd last column

awk 'NR==3 {print}' <file\_name> --> To display the 3rd row

----------------------------------------------------------------------------------------------------

Find --> To find files and directories in Linux

find -name <name> --> To search and display all the files and directories with the particular name

find -iname <name> --> Case Insensitive search

-type f --> To only search for files

-type d --> To only search for directories

find -mtime -10 --> To find and display all the files and directories created in the last 10 days

find -mtime +10 --> To find and display all the files and directories created more than 10 days ago

find -type f -perm 664 --> To search for all the files with the permission 664

find -type f -empty --> To search for all the empty files

find -maxdepth 1 -name <name> --> To only search for the current directory

Find the password file between sub-directory level 2 and 4.

**find / -mindepth 3 -maxdepth 5 -name passwd**

----------------------------------------------------------------------------------------------------

Link --> To create a shortcut for a file in linux

ln -s <original\_file\_path> <softlink\_name> --> To create a soft link

ln <original\_file\_path> <hardlink\_name> --> To create a hard link

A softlink will point to the path of the original file and once the file gets deleted/moved

The softlink will not work

A hardlink even after the original file gets deleted/moved The hardlink will still work

Reason: Hardlink points directly to the inode of the file

inode --> Is the unique identification number of file which points to the file's memory block

ls -i <file\_name> --> TO check the inode of a file

----------------------------------------------------------------------------------------------------

Miscellaneous

who --> To check all the users that are logged in to the server

whoami --> To check the current user

hostname --> To check the ip address of the system (Private)

curl ifconfig.me --> To display the public ip address

uname --> To check OS and details

uname -a --> To check all the details of the system

lsof → command list open files

----------------------------------------------------------------------------------------------------

tee - Redirect and Append

ls -l | tee <file\_name> --> To write the output of a command to a file and display the contents on the terminal

ls -l | tee -a <file\_name> --> For Append

----------------------------------------------------------------------------------------------------

Assignment:

1. Find all the non empty files

find -type f ! -size 0

2. Find all the files with size more than 1 MB

find <directory> -type f -size +N<Unit Type>

3. Find and delete all the empty files

find . -type f -empty -print -delete

----------------------------------------------------------------------------------------------------

Assignment:

Install Apache Tomcat Application By Tuesday

inside bin under root user

./startup.sh

./shutdown.sh

Requirements:

1. Java

sudo yum install <package\_name>

2. sudo yum install java

3. sudo yum install wget

3.1 wget URL

3.2 tar zxpvf apache-tomcat-7.0.42.tar.gz

3.3 set up the Tomcat users

**tomcat-users.xml** in **apache-tomcat-7.0.42/config** directory

3.4 to change port

conf sub-directory under Tomcat directory and open server.xml file

----------------------------------------------------------------------------------------------------

SSH - Secure Shell or Secure Socket Shell

SSH is a network protocol that enables users to access a server in a secure way over an unsecured network

Defualt Port:

SSH --> 22

Apache Tomcat --> 8080

Jenkins --> 8080

HTTP --> 80

HTTPS --> 443

DNS --> 53 (DNS uses TCP and UDP port number **53**)

Port: A port is a virtual point where network connections start and end

ssh <key> <username>@<ipadress> --> To SSH into a server

Example

ssh -i <PEM\_File> ec2-user@<ipadress> --> To SSH into a server using PEM file

SSH Passwordless Connection

ssh-keygen -t rsa --> To generate a rsa key pair (Public and Private Key)

id\_rsa --> Private Key

id\_rsa.pub --> Public Key

Copy the contents of the Public Key (id\_rsa) to the authorized\_keys file under .ssh folder

of the server that you want to establish password less connection.

Passwordless connection will be established

----------------------------------------------------------------------------------------------------

winscp --> To transfer Files

----------------------------------------------------------------------------------------------------

scp - To copy files over ssh

scp <file\_path> <user\_name>@<ipadress>:<destination\_path> --> To copy a file over ssh to another server

scp -i <pem\_file> <file\_path> <user\_name>@<ipadress>:<destination\_path> --> Without passwordless connection

rsync --> With rsync in case of any failures rsync will resume where the copy has stopped

----------------------------------------------------------------------------------------------------

Process (ps)

ps -ef --> To show all the running process on the server

ps -u <user\_name> --> To check all the process started by a particular user

kill/kilall --> To forcefully stop a process

kill -9 <PID> --> To kill a process by PID

killall -9 <process\_name> --> To kill a process by its name

killall -9 -u <username> --> To kill all the process started by an user

sudo service <process\_name> stop --> Gracefully stop a process

example:

sudo service docker stop --> To stop docker service

top --> To check all the process in an interactive view

----------------------------------------------------------------------------------------------------

Assignment --> Load Average in Linux

The load average is the average system load on a Linux server for a defined period of time. In other words, it is the CPU demand of a server that includes sum of the running and the waiting threads.

uptime command is used to see the load average

----------------------------------------------------------------------------------------------------

ping <hostname> --> To check and ping another server

ping www.google.com

ping 0 --> To ping current server

----------------------------------------------------------------------------------------------------

bashrc and bashprofile

These files execute everytime automatically as the session starts

----------------------------------------------------------------------------------------------------

Assignment: Find the difference between bashrc and bashprofile

To make an alias for the Terminal in OS X, you can either put the aliases in *.bash\_profile* or *.bashrc. .bash\_profile* is executed for login shells, while *.bashrc* is executed for interactive non-login shells.

When you login (type username and password) via console, either sitting at the machine, or remotely via ssh: *.bash\_profile* is executed to configure your shell before the initial command prompt.

But, if you’ve already logged into your machine and open a new terminal window (xterm) then *.bashrc* is executed before the window command prompt. *.bashrc* is also run when you start a new bash instance by typing */bin/bash* in a terminal.

----------------------------------------------------------------------------------------------------

We could setup environment variables and Alias for commands in these files

example

alias FED="find -type f -empty | xargs rm -rf"

bash\_history --> Stores all the commands that were executed on the server

----------------------------------------------------------------------------------------------------

uniq and sort

uniq <file\_name> --> To display only the unique values in a file

Drawback --> uniq command only works if the same values are adjacent to each other

sort <file\_name> --> To sort the values in a file

sort -r <file\_name> --> To sort in reverse order

sort <filename> | uniq --> To sort and eliminate all the duplicate values

----------------------------------------------------------------------------------------------------

netsat --> To check the information about ports in the linux server

netstat -a --> To check all the ports

netstat -l --> To check all that are in use

sudo netstat -tulnp --> To check which is using which port

----------------------------------------------------------------------------------------------------

& --> To run a command or a script in background

Syntax: command/script &

fg <PID/command/script> --> To bing the process/command/script to the foreground

----------------------------------------------------------------------------------------------------

Telnet:

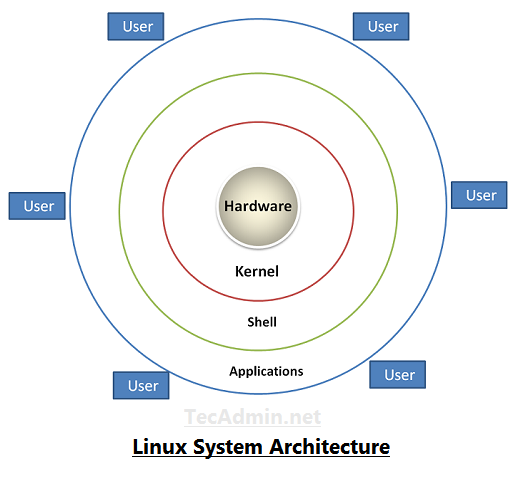
Default port: 23

Telenet is a networking protocol which is used to create a remote connection. The data being transferred

using this protocol is unencrypted

Syntax: telnet <hostname> <portnumber>

Linux Architecture



**Kernel**: is the core part of the Linux. It is responsible for all major activities of Linux OS. It consists of various modules and it interacts directly with the underlying HW

**Shell**: is an interface to the kernel which hides the complexity of the kernel’s functions from the users. It takes commands from the user and executes the kernel’s functions

**Types of shells:**

1. **Bourne shell:** the bourne shell is regarded as the first Linux/Unix shell ever. It is denoted as “sh”
2. **CShell (csh):** it was developed with the objective of achieving a scripting language similar to C-programming
3. **KShell (ksh):** it combines the feature of “Bourne shell” and “csh”
4. **Bourne Again Shell (bash):** it has all the features of bourne shell but it is much more efficient and easier to use. It supports additional features like piping, command substitution etc.

**To change Shell:**

chsh/lchsh <shell\_name> ----> to change current shell

echo $SHELL ----> to check current shell

--------------------------------------------------------------------------------------------------------------------------------------

**Bash Shell Scripting**

To perform repetitive tasks instead of running all the commands one by one, we can write these commands in a file and we can execute them. These files are called as shell scripts.

Extension of shell script is .sh

**Shebang**:

The first line of any shell script should always start with shebang

#!/bib/bash

Shebang invokes the bash shell and if it was not used, the shell script uses the default shell

**To execute shell scripts:**

1. ./<script\_name>.sh
2. Sh <script\_name>.sh
3. Bash <script\_name>.sh (for this you need executable permission to run the file)

**Variables**:

It is character string to which we assign some value. Value can be numbers, text, filename or any other data.

Name of variable can contain only letters, numbers and an underscore

To access the variable inside a shell script we have to use $ followed by the name of the variable

Examples:

1. Assigning var in text editor

#!/bin/bash

Name=”abc”

Place=”hyd”

Echo “my name is $Name. I am from $Place”

sh script\_name.sh to execute above script

1. Passing values during run time.

To pass arguments or values to shell scripts at run time, we use $1, $2,….${n}

#!/bin/bash

Echo “my name is $1. I am from $2”

sh script\_name.sh abc hyd to execute above script

**Special variables:**

$0 -----> the filename of the current script

$# -------> the total number of arguments passed to the script

$\* ------> to give all the arguments passed to script in string format

$@ -------> to give all the arguments passed to script in array format

$? ---------> it gives the status of last executed command (0 means successful, any other number represents last command isn’t executed properly)

$$ ------->to check the PID of the current running process

$! --------> gives the PID of last process that went into background

----------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------

26-01-2022 Class - 2

----------------------------------------------------------------------------------------------------

**Operators:**

1. Arithmetic Operators:

#!/bin/bash

sum=`expr $1 + $2`

sub=`expr $1 - $2`

echo "Sum of Two Numbers i $sum"

echo "Difference of Two Numbers is $sub"

----------------------------------------------------------------------------------------------------

1. Relational Operators:
2. Strings

Equal --> ==

Not Equal --> !=

Less Than --> <

Less Than or Equal --> <=

Greater Than --> >

Greatet Than or Equal --> >=

1. Numbers

Equal --> -eq

Not Equal --> -ne

Less Than --> -lt

Less Than or Equal --> -le

Greater Than --> -gt

Gteater Than or Equal --> -ge

----------------------------------------------------------------------------------------------------

**IF Condition:**

Syntax:

if [condition]

then

statements1

else

statements2

fi

------------------------------------------------

if [condition1]; then

statements1

elif [condition2]; then

statements2

else

statements3

fi

----------------------------------------------------------------------------------------------------

**Example:**

1. To find the biggest of two numbers

#!/bin/bash

first=$1

second=$2

if [ $first -gt $second ]; then

echo "$first is the biggest"

else

echo "$second is the biggest"

fi

----------------------------------------------------------------------------------------------------

#!/bin/bash

first=$1

second=$2

if [ $# -ne 2 ]; then

echo "Enter two numbers"

elif [ $first -eq $second ]; then

echo "The numbers are equal"

elif [ $first -gt $second ]; then

echo "$first is the biggest"

else

echo "$second is the biggest"

fi

----------------------------------------------------------------------------------------------------

**File Operators:**

[ -f $<name> ] --> To check if it is a filename

[ -d $<name> ] --> To check if it is a directory

[ -r $<name> ] --> To check whether the file has read permission

[ -w $<name> ] --> To check whether the file has write permission

[ -x $<name> ] --> To check whether the file has executable permission

[ -e $<name> ] --> To check whether the file exists

[ -s $<name> ] --> To check whether the file has some data

----------------------------------------------------------------------------------------------------

**Example:** Shell script to check a file or directory

#!/bin/bash

name=$1

if [ -f $name ]; then

echo "$name is a file"

if [ -r $name ]; then

echo "$name has read permission"

else

echo "$name does not have read permission"

fi

if [ -w $name ]; then

echo "$name has write permission"

else

echo "$name does not have write permission"

fi

elif [ -d $name ]; then

echo "$name is a directory"

else

echo "$name does not exist"

fi

----------------------------------------------------------------------------------------------------

**Debugging:**

set -x --> Prints commands and their arguments as they are executed.

set +x--> to disable debug.

set -e --> To stop a script immediately when a command exits with non zero status

set -t --> To exit after reading and executing only one command

----------------------------------------------------------------------------------------------------

Assignment: Shell Script to find the biggest of 3 Numbers

#!/bin/bash

a=$1

b=$2

c=$3

if [ $a -gt $b ] && [ $a -gt $c ]; then

echo "$a is big"

elif [ $b -gt $c ]; then

echo "$b is big"

else

echo "$c is big"

fi

Assignment: To check if the file has some data

#!/bin/bash

a=$1

if [ -f @a ]; then

echo "a is file"

if [ -s $a ]; then

echo "$a not empty"

else

echo "$a is empty"

fi

else

echo "$a is a directory"

fi

----------------------------------------------------------------------------------------------------

27-01-2022 Class - 3

----------------------------------------------------------------------------------------------------

Biggest of 3 numbers

#!/bin/bash

f1=$1

f2=$2

f3=$3

if [ $# -eq 3 ]; then

if [ $f1 -eq $f2 ] && [ $f1 -eq $f3 ] && [ $f2 -eq $f3 ]; then

echo "All Numbers are Equal"

elif [ $f1 -gt $f2 ] && [ $f1 -gt $f3 ]; then

echo "$f1 is greater"

elif [ $f2 -gt $f3 ]; then

echo "$f2 is greater"

else

echo "$f3 is greater"

fi

else

echo "Enter three numbers only"

fi

---------------------------------------------------------------------------------------------------

**While Loop**

while [condition]

do

statements

done

while [condition]; do

statements

done

----------------------------------------------------------------------------------------------------

Example: to find the sum on n numbers

#!/bin/bash

n=$1

sum=0

while [ n -gt 0 ]; do

sum=`expr $sum + $n`

n=`expr $n - 1`

done

----------------------------------------------------------------------------------------------------

Example: To find the factorial of a number

#!/bin/bash

n=$1

Factorial=1

while [ $n -gt 1 ]; do

Factorial=`expr $Factorial \\* $n`

n=`expr $n - 1`

done

echo "The Factorial of $1 numbers is $Factorial"

----------------------------------------------------------------------------------------------------

**While Loop To Read Lines**

Syntax:

while read line

do

echo $line

done < <file>

----------------------------------------------------------------------------------------------------

Example: To Print the conents of a file

#!/bin/bash

while read line

do

echo $line

done < $1

----------------------------------------------------------------------------------------------------

Example: While loop to check the number of characters in each line

#!/bin/bash

while read line

do

echo $line | wc -c

done < $1

----------------------------------------------------------------------------------------------------

Example:

Input:

This is Linux Red Hat Floavor

We are working in AWS CLoud

Output:

1: 20

2: 15

#!/bin/bash

i=1

while read line

do

wc=`echo $line | wc -c`

echo "$i:$wc"

i=`expr $i + 1`

done < $1

----------------------------------------------------------------------------------------------------

Example:

Name ID Age

abc 001 25

def 002 56

ghi 003 36

jkl 004 59

mno 005 29

Output: All the names of persons aged more than 40 years

def

jkl

#!/bin/bash

while read line

do

age=`echo $line | cut -d " " -f3`

if [ $age -gt 40 ]; then

name=`echo $line | cut -d " " -f1`

echo $name

fi

done < $1

----------------------------------------------------------------------------------------------------

Assignment: Script to add Even Numbers

Input:

n =5

Output:

Sum = 2+4 =6

#!/bin/bash

counter=$1

sum=0

i=0

while [ $i -lt $counter ]

do

sum=$(( $sum + $i ))

i=$(( $i + 2 ))

done

echo $sum

----------------------------------------------------------------------------------------------------

28-01-2022 Class - 4

----------------------------------------------------------------------------------------------------

Example: To Find the sum on Even Numbers upto n

1.

#!/bin/bash

echo "Enter the n value"

read n

sum=0

i=2

while [ $i -le $n ]

do

sum=`expr $sum + $i`

i=`expr $i + 2`

done

echo "Sum of even numbers is $sum"

2.

#!/bin/bash

echo "Enter the n value"

read n

sum=0

while [ $n -ge 2 ]

do

rem=`expr $n % 2`

if [ $rem -eq 0 ]; then

sum=`expr $sum + $n`

fi

n=`expr $n - 1`

done

echo "Sum of even numbers is $sum"

----------------------------------------------------------------------------------------------------

Example: Script to check the disk size

#!/bin/bash

size=`df -h | awk -F " " '{print$(NF-1)}' | sed -n "6p" | cut -d "%" -f1`

if [ $size -gt 85 ]; then

echo "The disk size is Full"

echo "Percentage Usage is $size"

fi

----------------------------------------------------------------------------------------------------

Example: Script to change the file extensions

t1.txt t2.txt t3.txt

t1.sh t2.sh t3.sh

#!/bin/bash

find -type f -name "\*.txt" > temp

while read line

do

name=`echo $line | sed s/.txt//g`

mv $line $name.sh

done < temp

rm temp

----------------------------------------------------------------------------------------------------

Example: Script to reverse a file

this is linux

we are working in AWS

output:

we are working in AWS

this is linux

#!/bin/bash

l=`cat $1 | wc -l`

while [ $l -gt 0 ]; do

head -$l $1 | tail -1 >> temp

l=`expr $l - 1`

done

cat temp

rm temp

----------------------------------------------------------------------------------------------------

Assignment: Script to Reverse a string

Input: Hello

Output: olleH

Hint: echo <string> | cut -c5 --> To cut the 5th letter in the string

echo "enter string"

read s

length=`echo $s|wc -c`

i=$((length-1))

while [ $i -gt 0 ]

do

wc=$wc`echo "$s"| cut -c$i`

i=$((i-1))

done

echo "$wc"

----------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------

29-01-2022 Class - 5

----------------------------------------------------------------------------------------------------

**Cron Job**

A Cron Job is a linux command used for scheduling tasks to be executed periodically

crontab --> This is a file which contains all the cron jobs entries

\* \* \* \* \* command/script

min hour date month day

00 - Sunday

01 - Monday

02 - Tuesday

03 - Wednesday

04 - Thursday

05 - Friday

06 - Saturday

5th of Feb Saturday on 6 PM --> 00 18 05 02 06 command/script

10 AM on every Monday --> 00 10 \* \* 01 command/script

10 AM on every Monday and Friday --> 00 10 \* \* 01,05 command/script

Every Hour everyday --> 00 \* \* \* \* command/script

Every 15 minutes on Monday --> \*/15 \* \* \* 01 command/script

crontab -e --> To edit the crontab file

crontab -l --> To list the existing crontabs

Syntax:

\* \* \* \* \* sh <Path\_to\_the\_Script>

----------------------------------------------------------------------------------------------------

**Mail Command**

sendmail, postfix, mailx

echo "content" | mail -s "subject" -c "cc" -b "bcc" <email\_id>

mail -s "subject" -c "cc" -b "bcc" <email\_id> < <file\_name\_path>

----------------------------------------------------------------------------------------------------

**For Loop**

for i in var1, var2 ......

do

statements

done

for i in {0..100} --> i will go from 1 to 100

for i in {0..100..2} --> i will go from 1 to 100 with increments of 2

for i in $\*

----------------------------------------------------------------------------------------------------

sh sum.sh 2 4 6 8 10

#!/bin/bash

sum=0

for i in $\*

do

sum=`expr $sum + $i`

done

echo "The total sum of numbers entered is $sum"

----------------------------------------------------------------------------------------------------

Example to find the factorial of n numbers

#!/bin/bash

for i in $\*

n=$i

Factorial=1

while [ $n -gt 1 ]; do

Factorial=`expr $Factorial \\* $n`

n=`expr $n - 1`

done

echo "The Factorial of $i is $Factorial"

done

----------------------------------------------------------------------------------------------------

systemctl is-active --quiet <service\_name> --> To check if a particular service is running

ps -C <service\_name> --> To check if a particular service is running

----------------------------------------------------------------------------------------------------

Example: A script to check if the services are down

#!/bin/bash

services="docker ansible terraform tomcat"

for i in $services

do

systemctl is-active --quiet $i

if [ $? -ne 0 ]; then

sudo systmctl start $i

echo $i >> stoppedservices

fi

done

mail -s "Services Stopped" abc@gmail.com < stoppedservices

----------------------------------------------------------------------------------------------------

Assignment: Write a script to change the file extensions from .c to .txt using for loop

----------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------

30-01-2022 Class - 6

----------------------------------------------------------------------------------------------------

**Functions:**

Syntax:

<functions\_name> ()

{

statements

}

----------------------------------------------------------------------------------------------------

Example: Hello Function

#!/bin/bash

hello ()

{

echo "Hello World"

echo "This has been printed inside the function"

}

echo "This is before we call the function"

hello

echo "This is after we call the function"

----------------------------------------------------------------------------------------------------

Example: Factiorial of a given set of numbers using functions

#!/bin/bash

fact ()

{

n=$1

Factorial=1

while [ $n -gt 1 ]; do

Factorial=`expr $Factorial \\* $n`

n=`expr $n - 1`

done

echo "The Factorial of $1 is $Factorial"

}

for i in $\*

do

fact $i

done

----------------------------------------------------------------------------------------------------

**Case Statements:**

Synatx:

case $variable in

pattern1) Statements to be executed if pattern1 matches the variable

;;

pattern2) Statements to be executed if pattern 2 matches the variable

;;

pattern3|pattern4) Statements to be executed if pattern3 or pattern4 matches the variable

;;

\*) Default statements to be executed if non of the patterns matches

;;

esac

----------------------------------------------------------------------------------------------------

Example:

#!/bin/bash

echo "Enter the Number"

read n

case $n in

1) echo "The number is 1"

;;

2) echo "The number is 2"

;;

3|4) echo "The number is 3 or 4"

;;

5|6) echo "The number is 5 or 6"

;;

\*) echo "Invalid number"

;;

esac

----------------------------------------------------------------------------------------------------

Example: To check the day

Monday --> Today is Monday

Tuesday --> Today is Tuesday

Wednesday or Thursday --> Today is either Wednesday or Thursday

Friday --> Today is Friday

Any other Imputs --> Saturday and Sunday are holidays

#!/bin/bash

echo "enter the day"

read day

case $day in

Monday) echo "Today is Monday"

;;

Tuesday) echo "Today is Tuesday"

;;

Wednesday|Thursday) echo "Toady is either Wednesday or Thursday"

;;

Friday) echo "Today is friday"

;;

\*) echo "Saturday and Sunday are holidays"

;;

esac

----------------------------------------------------------------------------------------------------

Example:

1 --> Search for files based on given input [find -type f -iname "<name>"]

2 --> Check if the file by the given input is present or not [ -e $<name> ]

3 --> Create a softlink [file name and also the softlink name] [ ln -s <original file> <sl name>]

#!/bin/bash

echo "1 --> Search for files based on given input"

echo "2 --> Check if the file by the given input is present or not"

echo "3 --> Create a softlink"

echo "Enter the Number"

read n

case $n in

1) echo "Enter the file name"

read name

find -iname $name

;;

2) echo "Enter the file name"

read name

if [ -e $name ]; then

echo "The file is present"

else

echo "The file is not present"

fi

;;

3) echo "Enter the orginal file name"

read original

echo "Enter the soft link name"

read softlink

ln -s $original $softlink

;;

\*) echo "Invalid Input"

;;

esac

----------------------------------------------------------------------------------------------------

**Environment Variables:**

export <KEY>=<VALUE>

Note: These Environment Variables will be removed after the session has been closed

**What is stickybit?**

A Sticky bit is a permission bit that is set on a file or a directory that lets only the owner of the file/directory or the root user to delete or rename the file. ... No other user is given privileges to delete the file created by some other user.

*other + t (sticky)*

The last special permission has been dubbed the "sticky bit." This permission does not affect individual files. However, at the directory level, it restricts file deletion. Only the owner (and root) of a file can remove the file within that directory

**How to check the kernel version?**

uname -v --> kernel version

uname -r --> kernel release

uname -s --> kernel name

**No Of cpu Connected To sever**

Nproc command

**what is shebang in linux**

It is called a shebang or a "bang" line. It is nothing **but the absolute path to the Bash interpreter**. It consists of a number sign and an exclamation point character (#!), followed by the full path to the interpreter such as /bin/bash. All scripts under Linux execute using the interpreter specified on a first line.

**How to check a particular running process?**

pgrep {process}

**how do you list volume block?**

lsblk -a

**xargs**

Pipe (|) is used when you want the standard output to be fed the next stage of the pipeline (cat foobar.txt | grep example).

xargs is used when you want to feed the standard output to be used as an argument to the next tool in the pipeline

(find . -name “foobar.\*” -print | xargs rm)

**Swap memory**

The swap space is located on disk, in the form of a partition or a file. Linux uses it to extend the memory available to processes, storing infrequently used pages there. We usually configure swap space during the operating system installation. But, it can also be set afterward by using the mkswap and swapon commands

**zombie process**

A zombie process is a process whose execution is completed but it still has an entry in the process table. Zombie processes usually occur for child processes, as the parent process still needs to read its child's exit status.

**Script to trigger emergency message on terminal and trigger mail when disk % reaches 75%**

**What is /bin?**

Bin is an abbreviation of Binaries. It's just a directory where a user of an operating system can expect to find applications. The different directories on a Linux system can be daunting or confusing if you aren't used to them. ... It contains essential binary files (unlike /usr/bin directory) also for booting.

**default user name for the AMI that you used to launch your instance:**

* For Amazon Linux 2 or the Amazon Linux AMI, the user name is ec2-user.
* For a RHEL AMI, the user name is ec2-user or root.
* For an Ubuntu AMI, the user name is ubuntu.

**Firewall**

* Is firewalld running on my system?

sudo firewall-cmd –state

* Stop the the firewalld

sudo systemctl stop firewalld

* Disable the FirewallD service at boot time

sudo systemctl disable firewalld

sudo systemctl mask --now firewalld

**disabling firewall in linux**

systemctl start firewalld

systemctl stop firewalld

--------------------------------------------------------------------------------------------------------------------------------------

#!/bin/bash

size=`df -h | awk -F " " '{print$(NF-1)}' | sed -n "6p" | cut -d "%" -f1`

if [ $size -gt 75 ]; then

echo "The disk size is full $size"

mail -s "disk is full $size" abc@mail.com

fi

----------------------------------------------------------------------------------------------------

**kernel tuning** --> modify the way kernel works by dynamically changing some of its parameters

kernel tuning --> sysctl

There are three ways to modify kernel tunables.

1. Using the sysctl command

2. By manually modifying configuration files in the /etc/sysctl.d/ directory

3. Through a shell, interacting with the virtual file system mounted at /proc/sys

----------------------------------------------------------------------------------------------------

**hypervisor -->**

A hypervisor, also known as a virtual machine monitor or VMM,

is software that creates and runs virtual machines (VMs). A hypervisor allows

one host computer to support multiple guest VMs by virtually sharing its

resources, such as memory and processing.

----------------------------------------------------------------------------------------------------

**Input Field Separator**

IFS is used to set field separator (default is while space).

The -r option to read command disables backslash escaping (e.g., \n, \t).

This is failsafe while read loop for reading text files.

while IFS= read -r field1 filed2 field3 ... fieldN

do

command1 on $field1

command2 on $field1 and $field3

..

....

commandN on $field1 ... $fieldN

done < "/path/to dir/file name with space"

----------------------------------------------------------------------------------------------------

**/etc**

folder contains all the configuration files of linux

----------------------------------------------------------------------------------------------------

**lan**

**wan**

**gateway**

**IP**

The Internet Protocol (IP) is a protocol, or set of rules, for routing and addressing

packets of data so that they can travel across networks and arrive at the correct destination

----------------------------------------------------------------------------------------------------

**connection protocols**

**TCP :**

TCP (Transmission Control Protocol) is an important network protocol

that lets two hosts connect and exchange data streams.

TCP guarantees the delivery of data and packets in the same order as they were sent.

----------------------------------------------------------------------------------------------------

**UDP**

User Datagram Protocol (UDP) is a communications protocol that is primarily

used to establish low-latency and loss-tolerating connections between

applications on the internet. UDP enables process-to-process communication,

while TCP supports host-to-host communication.

----------------------------------------------------------------------------------------------------

**logical volumes**

A Logical Volume is the conceptual equivalent of a disk partition

----------------------------------------------------------------------------------------------------

Red Hat Enterprise Linux - yum

Ubuntu - apt

centOS - yum

Fedora - dnf

Debian -apt

**Usermods in LINUX**

In it’s life span a process executes in [user mode and kernel mode](https://www.geeksforgeeks.org/dual-mode-operations-os/). The **User mode** is normal mode where the process has limited access. While the **Kernel mode** is the privileged mode where the process has unrestricted access to system resources like hardware, memory, etc.

**LVM**

In Linux, Logical Volume Manager (LVM) is a **device mapper framework that provides logical volume management for the Linux kernel**. Most modern Linux distributions are LVM-aware to the point of being able to have their root file systems on a logical volume.

**List Mounted Drives on Linux**

1. cat /proc/mounts
2. You can use [**mount** command](https://linoxide.com/how-to-mount-drive-in-linux/) to list mount points. When you run mount command without any options it will list mount points.

--------------------------------------------------------------------------------------------------------------------------------------

**is it legal to edit linux kernel?**

Yes. You can edit Linux Kernel because it is released under General Public License (GPL) and anyone can edit it. It comes under the category of free and open source software.