
LINUX

02-05-2022 Class - 1

Creating an Instance

Step1: Create an AWS account

Step2: Login and open EC2 Console]

Search for EC2 at the top of the AWS console

Step3: Create a RedHat Linux Server with t2.micro instance type

Configure Security Groups
Type - All Traffic
Source - Anywhere

Key Pair - Choose to create a new pair, Give a name and download the pem file
---
Connecting to the server

Step1: Download and install Putty

Step2: Open PuttyGen and create ppk file using the pem file downloaded from AWS

Puttygen --> Load --> Select PEM file (All Files) --> Save Private Key

Step3: Connect to the linux

Open Putty --> Paste the public IP address in the host Name

Under Connection --> SSH --> Auth --> Select the ppk File (generated using puttygen)

Login as ec2-user [Default username for RedHat ec2-user]

Linux Commands:

To create fike and directories:
mkdir <directory_name> --> To create a directory_name
touch <file_name> --> To create a file

To list the files:
ls --> To list all the files and directories
ls -l --> To list the files and directories in long format
ls -lt --> To list the files and directories sorted with time
ls -lrt --> To list the files and directories sorted with timein reverse order

To change Directory:
cd <directory_name> --> To change the directory
cd .. --> To go back to a previous directory
cd --> To goto home directory [/home/ec2-user]

To check the current directory:
pwd --> To check the present working directory

04-05-2022 Class - 2
---
vi - Text Editor

vi <file_name> --> To open the file for editing

Note: By default VI editor opens in command mode

esc --> To go back to command mode
i --> To enter insert mode in VI

esc + :w --> To save the file
esc + :q! --> To quit without saving the file
esc + :wq --> To save and quit from the file

w --> write
q --> quit
! --> Forcefully
---
cat <file_name> --> To check the contents of the file
tac <file_name> --> To display the contents of the file in reverse order
---
esc + :set nu --> To set the numbers for the file
esc + :set nonu --> To remove the numbers

Find and Replace strings

esc + :%s/<old_string>/<new_string>/g

% --> All the lines
s --> Substitute
g --> Globally [all the occurence of the pattern in the line]
ig --> Case insensitive replace

esc + 3s/<old_string>/<new_string>/g --> To make the replace only on the 3rd line
esc + 3,6s/<old_string>/<new_string>/g --> To make the replace only on the 3rd to the 6th line
esc + 3,$s/<old_string>/<new_string>/g --> To make the replace only on the 3rd to the end of the line
---
esc + dd --> To delete a line in VI
esc + 3dd --> Deletes 3 lines
---
esc + /pattern --> To search for a pattern in your file

note: Use n to go to the next occurence of the word
---
esc + :undo --> To undo a change
esc + :redo --> To redo a change

Remove the file and directories:

rm <file_name> --> To delete a file
rm -rf <directory_name> --> To delete a directory
rm -rf * --> Remove all the files and directories in the present location
---
Copy:

cp <file_name1> <file_name2> --> To copy a file
cp <file_name> <dir>/<file_name> --> To copy a file inside a directory
cp -r <dir1> <dir2> --> To copy a directory

Move:

mv <file_name1> <file_name2> --> To copy a file
mv <file_name> <dir>/<file_name> --> To copy a file inside a directory
mv <dir1> <dir2> --> To copy a directory

Echo --> Print

echo "<text>" --> To print the text on the terminal
echo -e "Hi \nHow are you" --> To print with new line using \n
---

05-05-2022 Class - 3

Redirect(>) and Append(>>)

echo "hello" > <file_name> --> Redirect the output of echo command to the file
echo "hello" >> <file_name> --> Append the output of echo command to the tip of the existing file
---
Word Count:

wc <file_name> --> The total number of lines, words and character present in the file
wc -l <file_name> --> The total number of lines present in the file
wc -c <file_name> --> The total number of characters present in the file
wc -w <file_name> --> The total number of words present in the file
---
Grep - Used to search for strings inside a file

grep "<pattern>" <file_name> --> To display all the lines with the matching pattern
grep -i "<pattern>" <file_name> --> Case insensitive search

grep -e "<pattern1>" -e "<pattern2>" --> To search for multiple patterns

grep -l "<pattern>" * --> To check for the pattern in all the files and display the names of the files
that the pattern is present

grep -l -R "<pattern>" --> To check recursively (i.e sub folders)

grep "^<pattern>" --> To display all the lines that start with the pattern
grep "<pattern>$" --> To display all the lines that end with the pattern

grep -c "<pattern>" --> To count the number of lines the pattern is present

grep -v "<pattern>" --> To display all the lines that does not have the pattern
---
File Size and Disk Size

df -h --> To check the disk size of the server
free -h --> To check the system memory of the server [RAM]

du -sh <file_name> --> To check the file size

SUDO - Super User Does

sudo <command> --> TO execute a command with root permission[when you face permission denied error]

sudo su --> To goto root user

Assignment:
1. Find the different package installers for different flavors of linux [Redhat, Ubuntu, Centos, Debian, Alpine]
2. Grep flag to check for all the files that does not have the pattern
---

06-05-2022 Class - 4

Add Users

sudo useradd <user_name> --> To add a user to the server
sudo passwd <user_name> --> To set the 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

To Give a user SUDO permissions

For an user to get Sudo permission, The user needs to be added to the sudoers file [/etc/sudoers]

sudo vi /etc/sudoers --> To edit the sudoers file

<user_name> ALL=(ALL) NOPASSWD: ALL

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 a group

To check all the groups in the server

getent group

cat /etc/group

Permissions in Linux

rw-rw-r--
rwxrwxr-x

r- Read Permission
w- Write Permission
x- Executable Permission
- --> No permission

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 --> 7
rw --> 6
x --> 1
- --> 0

chmod 664 <file_name> --> To give rw for owner, rw for group, r for others
chmod 777 <file_name> --> to rwx for owner, group and others
chmod 740 <file_name> --> To give rwx for owner, r for groups, no permission for others

chmod u+x <file_name> --> To give owner the executable permission
chmod o-w <file_name> --> To Remove write permission from the the others

u --> Owner
o --> Others
g --> Groups

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 of the file

chown <user_name>:<group_name> <file_name> --> To change owner and the group for the file
---
Defualt Permissions in Linux

Files --> 666
Directories --> 777

umask 002 --> to change the default permissions of the file and directories to 664 and 775
umask 022 --> to change the default permissions of the file and directories to 644 and 755

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

07-05-2022 Class - 5
---
head --> To print required starting number of lines

head -n <file_name> --> To display the starting n lines
head <file_name> --> To display the starting 10 lines
head -3 <file_name> --> To display the starting 3 lines
---
tail --> To print required lines from the bottom

tail -n <file_name> --> To display the last n lines
tail <file_name> --> To display the last 10 lines
tail -3 <file_name> --> To display the last 3 lines
---
pipe [|] --> To pass the output of one command to the next command

Example:

head -5 <file_name> | tail -1 --> Print the 5th line of the file
---
sed --> Stream Editor used to find and replace words and much more

sed 's/<old_string>/<new_string>/g' <file_name> --> To replace old string with new string
sed -i 's/<old_string>/<new_string>/g' <file_name> --> To replace old string with new string and make changes to the file

sed '2s/<old_string>/<new_string>/g' <file_name> --> To replace old string with new string in 2nd line
sed '2,4s/<old_string>/<new_string>/g' <file_name> --> To replace old string with new string from 2nd to 4th line
sed '2,$s/<old_string>/<new_string>/g' <file_name> --> To replace old string with new string from 2nd to end of file

Example: Developers dev (1) regularly install and build upon dev(2) third-party dev (3) dependencies
sed '1s/dev/test/1' file = 1 is for 1st occurance of dev in the line
sed '1s/dev/test/1' file = 2 is for 2nd occurance of the dev in the line where the dev(2) will replace by the (test)

sed '4d' <file_name> --> To delete the 4th line
sed '2,4d' <file_name> --> To delete the from 2nd to 4th line
sed '2,$d' <file_name> --> To delete the 2nd to end of the file

sed -n '2p' <file_name> --> To print the 2nd line
sed -n '2,4p' <file_name> --> To print the 2nd to the 4th line
sed -n '2,$p' <file_name> --> To print the 2nd to end of file
---
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 search for only files
-type d --> To search for only 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 file with permission 664

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

find -maxdepth 1 -name <file_name> --> To only search for the current directory
---
Assignment:
1. Find and delete all the empty files
2. Find all the files with size more than 1 MB
3. Find all non empty files
---
sudo gpasswd -M test,test3 multigroup

08-05-2022 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 only the 1st and 3rd column
cut -d " " -f2-4 <file_name> --> To display from 2nd to 4th column

d --> Delimiter
---
awk command

awk '{print}' <file_name> --> To display the contents of the file

awk -F " " '{print$1}' <file_name> --> To print the 1st column
awk -F " " '{print$1,$3}' <file_name> --> To print the 1st and 3rd column
awk -F " " '{print$NF}' <file_name> --> To print the last column
awk -F " " '{print$(NF-1)}' <file_name> --> To print the last but one column

awk 'NR==3 {print}' <file_name> --> To print the 3rd row
awk 'NR==2,NR==4 {print}' <file_name> --> To print from 2 to 4th row
---
Link --> To create a shortcut of a file in Linux

ln -s <original_file_path> <softlink_name> --> To create a softlink of the original file
ln <original_file_path> <softlink_name> --> To create a hardlink of the original file

A softlink will point to the path of the original file, Once the original file gets deleted/Moved, The softlink will not work

Whereas The hardlink points to the inode of a file, Even after deleting/Moving the original file.
The hardlink will still work

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

ls -i --> To check inode of files and directories

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 ip address]
curl ifconfig.me --> To check the public ip address

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

---

tee - Redirect and Append

echo "hello" | tee <file_name> --> To redirect the output of the command to the file and display the contents on the terminal

echo "hello" | tee -a <file_name> --> To append using tee command

---

09-05-2022 Class - 7

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
Nginx --> 80
HTTPS --> 443

ssh -i <key_file_path> <user_name>@ip_address --> To SSH into a server

Example:

ssh -i may2022.pem ec2-user@<ip_address>

SSH Passwordless connection:

To generate a key pair:
ssh-keygen -t rsa

.ssh Folder
id_rsa --> Private Key
id_rsa.pub --> Public Key

Copy the contents of the public key to the target server's authorized keys file present in .ssh folders

ssh <user_name>@<ip_address> --> After passwordless connection is established
---
SCP --> To copy files over ssh

scp -i <pem_file_path> <file_to_be_copied_path> <user_name>@<ip_address>:<destination_path> --> To Copy files using ssh
scp <file_path> <user_name>@<ip_address>:<destination_path> --> when passwordless connection is established

rsync --> With rsync in case of any failure while copying, rsync has the ability to 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/killall --> Forcefully stopping a process

kill -9 <PID> --> TO kill a process by its PID
killall -9 <process_name> --> To kill a process by its name
killall -9 -u <user_name> --> To kill all the process started by an user

sudo service <process_name> stop --> Gracefully stop a service

Example: sudo service docker stop

top --> To check all the process in the server with live updates
---
Assignment: Load Average in Linux
---
ping <hostname> --> TO check and ping another server
ex: ping www.google.com

ping 0 --> To ping current server
---
uniq and sort

uniq <file_name> --> To display only rhe unique values in a file

The drawback of uniq command is that it can work of the values are adjacent to each other

sort <file_name> --> To sort the values in a file
sort -r <file_name> --> To sort the values in reverse order

sort <file_name> | uniq --> To sort and eliminate all the duplicate values
---
telnet:

Defualt Port: 23

telnet <ip_address/hostname> --> To login to the server
telnet <ip_address/hostname> <port_number> --> If telnet is on any other port

Telnet is a networking protocol which is used to create a remote connection just like SSH but in
an unsecured manner. The data being transferred using this protocol is unencrypted
---
netstat --> To check the information about ports in the linux server

To install netstat --> sudo yum install net-tools

netstat -a --> To check all the ports
netstat -l --> To check all the ports in use

sudo netstat -tulnp --> To check which process is using whick tcp or udp port
---
Assignment:
Install Apache Tomcat on Linux Server

Requirements:
1. Install Java

sudo yum install <package_name>
sudo yum install wget

To check the contents of compressed packages
unzip -l <.zip_archive>
tar -ztvf <.targz_archile>
---

bashrc and bash_profile

These files execute everytime automatically as soon as the session starts

We could environment variables and Alias for commands in these files

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

USER="ec2-user"

bash_history --> It stores all the commands that were run on the server

Assignment: Difference between bashrc and bash_profile

SHELL

echo $SHELL --> To check the current shell
chsh/lchsh <shell_name> --> To Switch the shell

SHELL SCRIPTING

12-05-2022 Class - 1

BASH Shell Scripting:

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

The extension of shell scripts is .sh

To execute shell scripts:

1. ./<script.sh>
2. sh <script.sh>
3. bash <script.sh>

Shebang:

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

```
#!/bin/bash
```

Shebang invokes the bash shell and if it is not used, the shell script will use the default shell

---

13-05-2022 Class - 2

Variables:

Varibales is a character string to which we can assign some value, The value can be a number, text,
filename or any other data

The name of the varibale can only contain letters, numbers and underscore

To access the value of the variable inside the script we have to use the "$" followed by the name
of the variable

Example:

1. Assigning values inside the script

```
#!/bin/bash
name="abc"
place="Bengaluru"
echo "Hi How are you"
echo "This is $name, I am From $place"
```

Output:

```
Hi How are you
This is abc, I am from Bengaluru
```

---

1. Passing the values during the run time

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

```
#!/bin/bash

echo "Hi How are you"
echo "This is $1, I am From $2"
```

Output:

```
Hi How are you
This is abc, I am from Bengaluru
```

While executing the script sh <script.sh> abc Bengaluru

Special Variables:

$0 --> The filename of the current script
$# --> The total number or arguments passed to the script

$* --> Gives all the arguments passed to script in string format
$@ --> Gives all the arguments passed to script in array format

$? --> TO check the status of last executed commands
$$ --> To check the PID of the current running process
$! --> To check the PID of the last process that went into background

---

Operators:

1. Arithmetic Operators [+, -, /, *]

Example:

```
#!/bin/bash

num1=$1
num2=$2

sum=`expr $num1 + $num2`
sum2=$(($num1 + $num2))

mult=`expr $num1 \* $num2
mult2=$(($num1 * $num2))

echo "The sum of two numbers is $sum - $sum2"
echo "The product of two numbers is $mult - $mult2"
```

---

2. Relational Operators

a. Strings

Equal --> ==
Not Equal --> !=
Less Than --> <
Less Than or Equal to --> <=
Greater Than --> >
Greater of Equla to --> >=

b. Numbers

Equal --> -eq
Not Equal --> -ne
Less Than --> -lt
Less Than or Equal to --> -le
Greater Than --> -gt
Greater of Equal to --> -ge

IF Condition

```
if [condition]
then
        statements
else
        statements
fi

if [condition]; then
```

```
        statements
else
        statements
fi
```

Example:

1. To find the biggest of 2 numbers:

```
#!/bin/bash

num1=$1
num2=$2
if [ $num1 -gt $num2 ]; then
       echo "$1 is the biggest"
else
       echo "$2 is the biggest"
fi
```

```
#!/bin/bash


num1=$1
num2=$2

if [ $# -ne 2 ]; then
        echo "Please enter 2 Numbers"
elif [ $num1 -eq $num2 ]; then
        echo "Both the numbers are equal"
elif [ $num1 -gt $num2 ]; then
        echo "$1 is the biggest"
else
        echo "$2 is the biggest"
fi
```


14-05-2022 Class - 3

File Operators:

[-f $<file_name>] --> To check whether the given input is a file
[-d $<dir_name>] --> To check whether the given input is a Directory

[-r $<file_name>] --> To check whether the given filename input has read permission
[-w $<file_name>] --> To check whether the given filename input has write permission
[-x $<file_name>] --> To check whether the given filename input has execute permission

[-e $<file_name>] --> To check whether the given filename input exists
[-s $<file_name>] --> To check whether the given filename input has some data
---
Example: Shell script to check a file or directory

```
#!/bin/bash

echo "Enter the name"
read name

if [ -f $name ]; then
        echo "The $name is a file"
elif [ -d $name ]; then
        echo "The $name is a directory"
else
        echo "$name doesnot exist"
fi

(or)

if [ -f $name ]; then
    echo "The $name is a file"
    if [ -r $name ]; then
        echo "The file has read permission"
    else
        echo "The file does not have read permission"
    fi
    if [ -w $name ]; then
        echo "The file has write permission"
    else
        echo "The file does not have write permission"
    fi

elif [ -d $name ]; then
    echo "The $name is a directory"
else
    echo "$name doesnot exist"
fi
```

Debugging:

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

---

While Loop

syntax:

```
while [condition]
do
        statements
done

while [condition]; do
```

```
        statements
done
```

Example: To find the sum of n numbers

```
#!/bib/bash

echo "Enter the n value"
read n
sum=0
while [ $n -gt 0 ]; do
        sum=`expr $sum + $n`
        n=`expr $n - 1`
done
echo "The sum of all the numbers is $sum"
```

Example: To find the factorial of a number

```
#!/bin/bash

n=$1
fact=1
while [ $n -gt 1 ]
do
        fact=`expr $n \* $fact`
        n=`expr $n - 1`
done

echo "The factorial of $1 is $fact"
```

Assignment:
1. The file operator the check whether the input is a symbolink link
2. Shell script to get the biggest of 3 numbers

15-05-2022 Class - 4

Example: Find the greatest of 3 numbers

```
#!/bin/bash/

num1=$1
num2=$2
num3=$3

if [ $# -ne 3 ]; then
        echo "Enter three numbers"
elif [ $num1 -eq $num2 ] && [ $num1 -eq $num3 ]; then
        echo "All the numbers are equal"
elif [ $num1 -ge $num2 ] && [ $num1 -ge $num3 ]; then
        echo "$num1 is greatest of all"
```

```
elif [ $num2 -ge $num3 ]; then
        echo "$num2 is greatest of all"
else
        echo "$num3 is greatest of all"
fi
```

Syntax : While Loop to Read a file line by line

```
while read <variable>
do
        echo $<variable>
done < <file_name>
```

Example: To print the contents of the file
cat <file_name> --> TO print the contents of the file

```
#!/bin/bash

while read line
do
        echo $line
done < $1
```

Contents of File:
this is linux
we are writing shell scripts

1st iteration

line="this is linux"

2nd Iteration
line="we are writing shell scripts"

Example: While Read loop to check the number of character in each line of the file

```
#!/bin/bash

while read line
do
        echo $line | wc -c
done < $1
```

Example: To get the number of characters in each line with line numbers

Input:
this is linux
we are writing shell scripts

Output:
1: 15

2: 25

```
#!/bin/bash

i=1
while read line
do
    wc=`echo $line | wc -c`
    echo "$i: $wc"
    i=`expr $i + 1`
done < $1
```

Example: Write a script to get all the names of employees with age greater than 30

Input:

```
Name ID Age
abc 001 35
def 002 26
ghi 003 54
jkl 004 22
mno 005 35
```

Output

```
abc
ghi
mno
```

```
#!/bin/bash
cat $1 | sed 1d > temp
while read line
do
        age=`echo $line | cut -d " " -f3`
        if [ $age -gt 30 ]; then
                name=`echo $line | cut -d " " -f1`
                echo $name
        fi
done < temp
rm temp
```

(or)

```
#!/bin/bash

i=1
while read line
do
                if [ $i -ne 1 ]; then
                        age=`echo $line | cut -d " " -f3`
                        if [ $age -gt 30 ]; then
            name=`echo $line | cut -d " " -f1`
```

```
                echo $name
                        fi
                else
                        i=`expr $i + 1`
done < $1
```

Example: Script to change the file extensions

t1.txt t2.txt t3.txt
t1.py t2.py t3.py

```
#!/bin/bash

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

while read line
do
        name=`echo $line | sed s/.txt//g`
        mv $line $name.py
done < temp
rm temp
```

(or)

```
#!/bin/bash

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

while read line
do
        name=`echo $line | sed s/.txt/.py/g`
        mv $line $name
done < temp
rm temp
```


16-05-2022 Class - 5

Example: Script to dispaly contents of a file in reverse order
tac <file_name> --> Display to contents in reverse order

Input:
this is linux
we are writing shell scripts

Output:
we are writing shell scripts
this is linux

```
#!/bin/bash
```

```
n=`wc -l $1`
while [ $n -gt 0 ]
do
  sed -n "$n"p $1
  n=`expr $n - 1`
done
```

(or)

```
#!/bin/bash

n=`wc -l $1`
while [ $n -gt 0 ]
do
        head -$n $1 | tail -1 >> temp
        n=`expr $n - 1`
done
cat temp
rm temp
```

Assignment: Reverse a string

Input: Hello
Output: olleH

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

---

Example: Script to check the disk usage

```
#!/bin/bash

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

if [ $size -gt 80 ]; then
        echo "The disk size is Full"
        echo "Percentage usage is $size"
fi
```

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 entries

```
*               *               *               *               *               command/script
min             hour    date    month  day
```

00 - Sunday
01 - Monday
02 - Tuesday
03 - Wednesday

04 - Thursday
05 - Friday
06 - Saturday

20th Feb Saturday 0n 2PM --> 00 14 20 02 06 command/script
4PM on every Wednesday --> 00 16 * * 03 command/script

Every Hour Everyday --> 00 * * * * command/script
Every 15 Minutes --> */15 * * * * command/script
Every Minute --> * * * * * command/script

crontab -e --> To edit the crontab file
crontab -l --> To list the existing crontabs

Syntax:

* * * * * sh <path_of_the_shell_script>

Mail Command:

sendmail, postfix, mailx

echo "content" | mail -s "subject" -c "cc" -b "bcc" -a "<attachment_file_path>" <email_id>
mail -s "subject" <email_id> < <file_name>

For Loop:

Syntax:

```
for i in var1, var2 .. ... ....
do
        statements
done
```

for i in {0..10} --> i will fo from 0 to 10
for i in {0..50..2} --> i will go from 0 to 50 with increments of 2

for i in $* --> i will take all the arguments passed to the script

---

Example: Sum of all the numbers passed to the script

sh sum.sh 2 3 8 10 7

```
#!/bin/bash

sum=0
for i in $*
do
        sum=`expr $sum + $i`
done
echo "The total sum of numbers is $sum"
```


16-05-2022 Class - 5

Example: To reverse a string

```
#!/bin/bash

echo "enter the string"
read string
n=`echo $string | wc -c`

while [ $n -gt 0 ]
do
   echo $string | cut -c$n >> temp
   n=$(($n-1))
done
cat temp | tr -d '\n'
rm temp
```

(or)

```
#!/bin/bash

echo "enter the string"
read string
n=`echo $string | wc -c`

while [ $n -gt 0 ]
do
   t=$t`echo $string | cut -c$n`
   n=$(($n-1))
done
```

Example: Find the factorial of n numbers

Input: 2 3 4

Output:

The factorial of 2 is 2
The factorial of 3 is 6
The factorial of 4 is 24

```
#!/bin/bash

for i in $*
        n=$i
        fact=1
        while [ $n -gt 1 ]
```

```
	do
		fact=`expr $n \* $fact`
		n=`expr $n - 1`
	done
	echo "The factorial of $i is $fact"
done
```

---

ps -C <service_name> --> To check if a particular service is running
systemctl is-active --quiet <service_name> --> To check if a particular service is running

---

Example: A script to check if the service is down

```
#!/bin/bash

services="docker jenkins ansible"
for i in $services
do
	systemctl is-active --quiet $i
	if [ $? -ne 0 ]; then
		sudo systemctl start $i
		echo $i >> stoppedservices
	fi
done
mail -s "Stopped Services" abc@gmail.com < stoppedservices
rm stoppedservices
```

Functions:

Syntax:

```
<function_name> ()
{
	statements
}
```

Example: Hello World Function

```
#!/bin/bash

hello ()
{
	echo "Hello World"
	echo -e "Printed inside the function \n"
}

echo -e "This is printed before calling the function \n"
hello
echo -e "This is printed after calling the function "
```

Example: To find the factorial of n numbers using functions

```
#!/bin/bash

fact ()
{
        n=$1
        fact=1
        while [ $n -gt 1 ]
        do
                fact=`expr $n \* $fact`
                n=`expr $n - 1`
        done
        echo "The factorial of $1 is $fact"
}

for i in $*
do
        fact $i
done
```

Case Statements:

Syntax:

```
case $variable in
pattern1) Statements if pattern1 matches the variable
;;
pattern2) Statements if pattern2 matches the variable
;;
pattern3|pattern4) Statements if pattern3 or pattern4 matches the variable
;;
*) Default statements to be executed if non of the pattern matches the variable
```

Example: To check for a particular number

```
#!/bin/bash

echo "Enter the number"
read n
case $n in
1) echo "The number is 1"
;;
2) echo "The number is 2"
;;
3) echo "The number is 3"
;;
4|5) echo "The number is 4 or 5"
;;
*) echo "The number is invalid"
;;
esac
```

Example:

1 --> Search for files based on given input
2 --> Check If a file is present based on given input
3 --> Create a softlink
4 --> Create a hardlink

```
#!/bin/bash

echo "1 --> Search for files based on given input"
echo "2 --> Check If a file is present based on given input"
echo "3 --> Create a softlink"
echo "4 --> Create a hardlink"

echo -e "\n Enter the number"
read n

case $n in
1) echo "Enter the file name"
read name
find -type f -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 path of original file"
read original
echo "Enter the path of softlink"
read softlink
ln -s $original $softlink
;;
4) echo "Enter the path of original file"
read original
echo "Enter the path of hardlink"
read hardlink
ln $original $hardlink
;;
*) echo "Invalid Input"
;;
esac
```

Environment Variables:

export <key>=<value>

Note: These environment are session specific.

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

Syntax: command/script &

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

---

loadaverage 0.4 1 5

60% free in last 1 minute

100% in use in last 5 minutes

400% overloaded in last 15 minutes
