**Shell Script Docs**

LINUX COMMANDS

# head -10 file ---- it shows first 10 lines

# tail -10 file ----- it shows last 10 lines

For eg in my file there are 100 lines I want to show middle 40 to 50 (10 lines)

Using pipe operator “|”

Combine two commands using pipe operator

# cat file | head -50 file | tail -10 file this will show last 10 lines from 40 to 50

Pipe operator combine the two commands that is it takes the first commands output as input to second commands.

# wc –l file ---word count in line commands

# wc –c file – check how many characters in file

# wc –w file –check how many words in file

---Search particular word or pattern in file----

GREP – global regular expression print --searches

# cat /var/log/message | grep warning

It will search the warning in that file

# cat /var/log/message | grep –i network –it will ignore the case sensitive (search both network and Network, nEtwork etc)

# cat /var/log/message | grep –w network search the particular work word only

Grep –n for check line numbers

If we want to search the word before two lines and after the two line occurrence

# cat file | grep -A2 abc – print n numbers after occurence

# cat file | grep -B2 abc -- print n numbers before occurrence

####one dynamic application is continuously sharing the log and how to check the logs####

# tail –f /var/log/messages – ctrl+c to quit >>> it shows logs dynamically

###############Editors######

Vi

Vim

Nano

Gedit

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ \_\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

Vi – esc mode/command mode , and insert mode

To write something – insert mode and execute command – in esc mode

To save :wq and quit !q

“yy is to copy and p is paste in file dd is cut”

“3yy will copy 3 lines yard”

In file -- /word will go to the word in file

#### copy lines from multiple files#####

Vim -O file1 file2

It will open the 2 files horizontally in display and to switch between two files” ctrl+w”

###files.sh######

Vim new.sh --- color syntax

#! /bin/bash

While true

{

do

echo “Hello”;

done

}

Delete lines “dd” single character x

########## SHE-BANG#########################

#! /bin/bash ---interpreter

1. First and foremost line
2. Comments

#! /bin/bash

echo Hello

####### COMMAND-NAME {options} {input} ####

Command may or may not have option and input….

echo {option} option

-e - Enable esc seq - “\n new line, \t new tab, \e new color”

Every commands have a process ID and make kernel busy to resolve this reduce the code size and less number of commands

# echo –e “hai\nbye” – double and single quotes need to use

# echo –e “hai\tbye” - \t is tab space

###### check the colours

# man console\_codes

#########Using color in print commands

# echo –e “\e\_[31\_m\_input” no space requires

# echo –e “\e[31mhello” – it will print in red color each colors having numbers—red 31 green 32…

###using both foreground and background

# echo –e “\e[1;33;40mhai” number 1 is bold and 33 is background and 40 is foreground.

# echo –e “\e[1;31;40mhello\e[0m” -- this will onlyenable colors in particular output not all.

###### echo –n ------#########

##### -n option will use to avoid the new lines####

#echo –n hai

#########REDIRECTORS#############

Keyboard-🡪 input 🡨-- file

Term 🡨- output -🡪 file

Input and output is redirectories

STDIN <

STDOUT >

###example using – mail command

# mail –s test1 root@localhost <message

Message is a file which store the content for mail and it will easyily tracked

### < std in input is using#####

#### output comprises of two things 1. Std out 1> and 2. Stderr 2>

# ls >out

# ls –l new.sh new1.sh 1>out 2>err

What is this new.sh file is there but new1.sh is not in the system now new.sh output is saved in 1> out means saved in out and the error saved in 2>err err file.

###& will save both output and error in same file

# ls –l new.sh new1.sh &>out

# ls file >>out -- >> is used to avoid overwrite

# ls | tee out

# ls | tee –a out after that check with cat out it will shows double time of output…

# ls –l new.sh new1.sh >out 2>&1 – this is also show the output in same file…

######/dev/null – is the file which can be the trash place in OS when you map any files here it will trash

# ls –l new.sh new1.sh >/dev/null

#### Shell Script wont stop if one commands fail this is the major drawbacks####

####For Example --- first I want to exe useraadd test then set passwd for user all are continuously eexecuting one by one even if its fails#######

###To avoid this use Exit status: 0-255

1. Success 1-255 –unsuccess partially ? is a variable

Access variable using $ symbol

#useradd user1 &>/dev/null

# echo $? ---127 means previous commands fails

#### 1 – input error , 2- command error, 126 –permission denied, 127 – command not found, 128+n – kill signals

###total kill signals ---64

#kill –l

####Admin uses 1, 2 ,9 15 kill processs####

Example #sleep 500 by other user

#ps –ef | grep sleep

Then find the p ID of sleep then I kill it

#kill -9 45777

Then user can find echo&? It shows 137 that is 128+n

####PIPE symbol###

Use for combine the commands

The first command output will converted to input.

##STDOUT------------------🡪>STDIN####

com1>out

com2<out

#cat /etc/passwd | grep root or use this for same output

# cat /etc/passwd >out

# grep root >out

#####Mail and chat###

#wall is the command send to everyone

###mutt is used to attach the file in mail####

##########QUOTES######################################################

##special characters like \*

# echo \* ---which will show the lists

## disable the special characters by using backslash, single quotes,double quote…

### \, ‘’, “”

### command quote ``

# echo Today date is `date`

####Double quotes – using two charaters -- $ - access variables and ` - execute commands

####/ is used to hide the special charcters

# echo –e “hai\nbye” in case if I want to hide the special characters \n by hide using another \ like

# echo –e hai\\nbye

########### TEXT FILTERS######################################

1. Line number
2. Row based
3. Column based

# cat –n /var/log/messages

1. Line numbers – head and tail

# cat –n /var/log/message | head –n 15

# cat –n /var/log/message | wc –l

#cat –n /var/log/messages | head –n 1599 | tail -11

#####Row based#####

###Grep command – used for row based

# head /etc/passwd >/tmp/passwd – what it will do the head command copy first 10 lines from /etc/passwd and move it to tmp.

# type grep – highlight the color

# /bin/grep –e root –e adm /tmp/passwd –color=auto

# grep –E ‘root|adm” passwd

####In case of scenario if I want to search more than 2 to 200 words will be possible using grep commands like grep –E ‘rrot|a|b|c|d|e|….|’ like this if I want to find more more means then use the below

###create one file search.grep

And store what are the words need to find then you can find it

# grep –f search.grep passwd

###single word from multiple files

# grep root /etc/passwd /etc/group –h

# grep root passwd –c

# grep -i abc patterns – case sensitive

# grep –x abc patterns

####sort commands

# cat number.txt | sort –n –r

# cat number.txt | sort –n |uniq – to avoid the duplicate

# cat number.txt | sort –n | uniq –d only shows duplicate

####empty lines are denoted with ^$ symbol

###6 symbol starting with line and $ symbol end with file###

#grep ^root /etc/passwd – will show the starting lines from root

#grep bash& /etc/passwd – will show the ending lines with bash

###show which is not having the words

# grep root /etc/passwd –v – will show the other lines except root

####Filter the output through column using cut cmd###

###Delimiter is delimits the column ###

###first im creating two column and row####

# echo –e “a\tb\tc\td\n1\t2\t3\t4”

#echo –e “a\tb\tc\td\n1\t2\t3\t4” | cut –f1,3 – will show first and third column

### when some files having : in the column for that using####

# cat /etc/passwd | cut –f 1 –d :

#echo HelloWorld | cut –c 1-5 –will cut the characters and show the output

###mostly cut cmds cannot find the if delimits is not there at that time its difficult to use cut for that we can use the command xargs

###xargs can convert the rows to column

#echo –e “1\n2\n3”

#echo –e “1\n2\n3” | xargs

####show free memory only

# free –m | grep Mem | xargs | cut –d ‘ ‘ –f 4

#### we are using three commands for this output for this we can use alternate command called awk

###xargs command will helpful to give stdin for the pipe operator for std out

# echo err | rm –f – it will not delete the file

# echo err | xargs rm –f this will delete

###########awk##############################

### $1 $2 $3 $4 …… $NF

Last field is nth field

Awk command print $1

Syntax of awk – awk ‘{print $!}’

Delimiter --- field to field in cut command tab

1. Single space
2. Multiple spaces
3. Tab spaces

# echo –e “a b c\t\td”

# free –m | grep Mem | awk ‘{print $4}’

### to search using awk using / / in words

# awk ‘/root/ {print}’ /etc/passwd

### -F stands for field separator

# awk –F : ‘ {print $1,$NF}’ /etc/passwd

#free –m | awk ‘/Mem/ {print $4}’

###### sed –

####sed syntax --- sed –e ‘comm1’ -e ‘com2’ …. File you will get output#############

####sed –i –e ‘com1’ file ---- you will not get output#####but it will apply in file

#### sed replaced by ed in old version ed is used####

P –print , d – delete , s- substitute, a- append, i- insert, c- change

### sed [option] {sed-cmds} {input files}

##sed [option] –f {sed cmds in a file} {input file}

## sed –f sed-file. Commands will be in the files

## -n will disable the duplicate lines####

# sed p emp.txt

101.sarvan,it

102,new,job

103,azure,cloud

104,aws,new

105,linux,all

106,devops,new

##################

#sed –n p emp.txt –to avoid duplicate

# sed –n –e ‘/linux/ p’ –e ‘/devops/ p’ emp.txt

# sed –n ‘4 p’ emp.txt – will show the 4 th line

### create one sed file and type the commands like

/linux/ p

/devops/ p

Then call the file through cmd in f

# sed –n –f new.sed emp.txt

## , will show 1 to 4 lines

# sed –n ‘1,4 p’ emp.txt

# sed –n ‘1~2p’ emp.txt will print the odd numbers of lines

# sed –n ‘/sarvan/,+2 p’ emp.txt

# sed –n ‘/saran/, +3 p’ emp.txt --- if the first is not found then it wont find the next lines

### delete commands in sed#####

# sed ‘/sarvan/ d’ emp.txt – it will delete the line --but itwill show in output not delete in file for that use -i

###delete empty line using ^$###

#sed -e ‘/^$/ d’ emp.txt

# sed –i –e ‘/^$/ d’ emp.txt

######substitute cmd###

Now in the sceanario I want to delete the particular word in the file ex linux in the file

# sed –e ‘9 s/linux//’ emp.txt

Now I want to substitute the words in lines using sed

# echo apple ant | sed –e ‘s/a/A/’

#echo apple ant | sed –e ‘s/a/A/g’ – globally changed all the words in the line

To avoid case sensitivity in substitiute use i

# sed –s ‘s/Sarvan/Saravanan/i’ emp.txt

##### Add new lines in file using SED####

Append – A

# sed –i ‘2 a 203,sara,engg’ emp.txt

# sed ‘$ a 203,sara,engg’ emp.txt -- $a will create new line after end of the line..

#####Insert content in already line####

# sed ‘2 i 203 sara,engg’ emp.txt –

## Change command

This will modify the line by changing the lines -- c

# sed –i –e ‘/^PasswordAuthentication/ s/no/yes/’ /tmp/sshd\_config

# sed –I –e ‘/PasswordAuthentication/ c PasswordAuthentication yes’ ssh\_config

#######VARIABLES#########################################

###If u assign a name to a set of data##########

###bash syntax ----- name=data

Ex – NoOfOranges=10

Data can be

Single character (a-z, A-Z, 0-9,\_)

Single word

Single line

Multiple lines

No data types --- input is consider as string

“” ‘’ -- to restrict the special characters

No data limitation

Name=Data

$Name ---$varaiable name

###Example########

Name=Data

#echo $Name

Variables are stored in memory

Using Process ID

####command called unset varaiable name

#unset Name ( Name is variable name)

For example when we try to create on 3gb file it will take some time but when we try to remove the file using rm cmd it takes one second to delete but its not delete it completely,

# time dd if=/dev/mapper/rl-root of=dddfile bs=1M count=2000

## unlink command is to remove the file######## file system will have inode tables which has used and unused part

## rm commands will send the file from used inode to unused inodes…

###Lets write simple variable script

#!/bin/bash

R=”\e[31m”

B=”\e[34m”

YOR=”\e[33;41m”

N=”\e[0m”

echo –e “$R Red Color $N”

echo –e “$B Blue Color $N”

#########Create one script to create user and passwd

username=test

useradd $username

echo `date | md5sum | cut –c 1-6` | passwd –stdin $username

echo User Created

echo “username and passwd = $username :: `date|md5sum|cut –c 1-6`”

chage –d 0 $username

### Properties of the variables#####

1. Read Write -🡪 Readonly

If I Want to make the varaiable for readonly

#readonly name

We cant set back readonly variable as readwrite

#umask

Umask value is set and able to get permission for the created files

#id

Every users have user id

Uid for root user is zero(0)

1. Local properties

For example create one name=sarvan in normal cmd

# name=sarvan

# echo $name

It will show the output but in the same terminal when create one .sh script as

#! /bin/bash

echo $name and try to execute it will not show the output

when the script tried to create new pid and try to get the output but its not there that is the reason it shows empty in output…

Every Process ID will get a parent process ID.

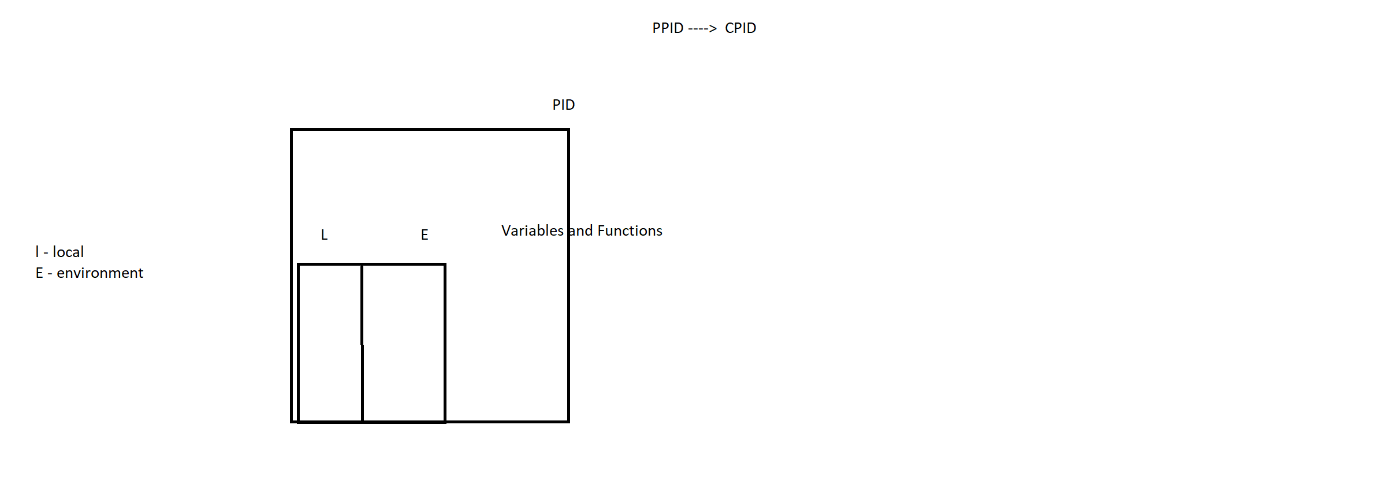
#ps

#pstree –p

Ppid🡪 cpid --- parent and child procee ID

Shift some thing from one to another using export

#export name



Every variables having two local and environment in parent pid the environment variable will get to child environment so for that environment we are using export commands.

Local-environment ----export commands

How import happens

Export will always access from parent to child for access child to parent use dot command

# script --- export name1=sarv

When exe this it will not get then also echo $name1 it will not show because pid is closed.

# . script.sh or source script will show the output. (here .dot is not path) ---its actually import

# enable –it is a command to show built shell commands

#### Scalar –properties

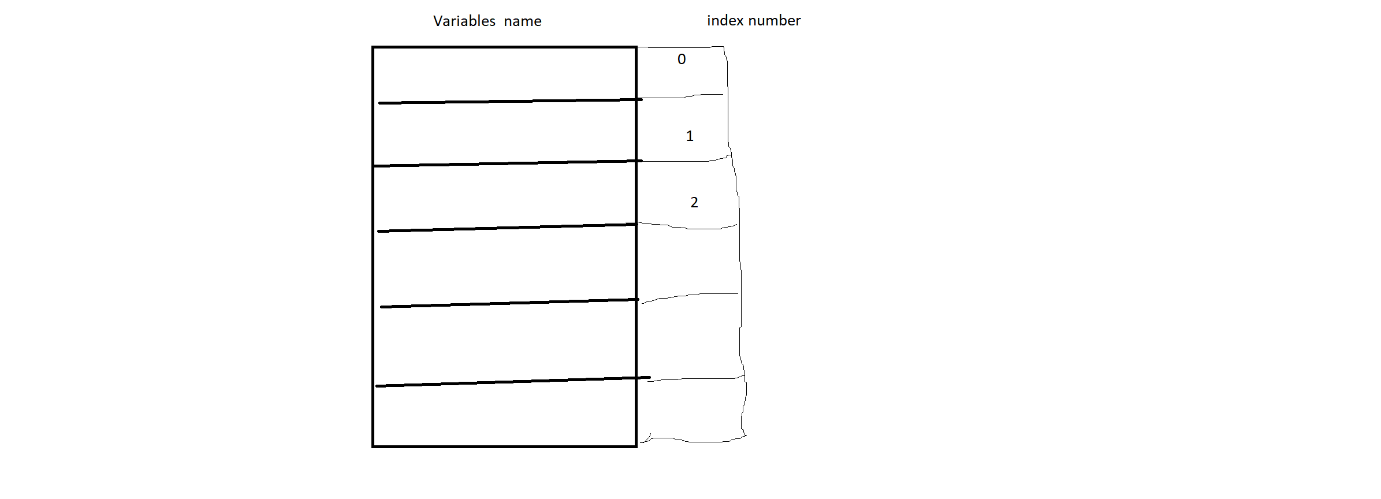
At any point of time we can access one value

Example name=Raghu

# echo $name ---output Raghu f I change the name as ramu then it will shows ramu its overwrite

To store multiple values using ARRAY

Array is using index number for each variables



This varaibles and index number will store the data

# student[0]=sarvan

#echo ${student[0]}

To get all the values

#echo ${student[\*]} \* or @ can show all values.

To get how many numbers of values

#echo ${#student[\*]} add # before the values it will show how many numbers of values

Shows 2 values (sarvan and sarv)

Example array

#!/bin/bash

Student= (1 3 4 5 6)

echo ${student[0]}

echo ${student[1]}

echo ${#student[\*]}

echo ${student[\*]:2:4}

## we can define varaiables in two ways

1. Command substution

VAR=$(commands)

VAR=`commands`

Example #!/bin/bash

DATE=$(date +%F) this will show format of the date

1. Arithematic substation

VAR=$(( arth eq ))

ADD=$(( 1+2 ))

#echo $ADD

Example script for calculate free memory % with arithmetic substation

#!/bin/bash

Total=$(free –m | awk ‘/^Mem/ {print $2}’)

Avail=$(free –m | awk ‘/^Mem/ {print $4}’)

Per=$(($Total\*100))

Per=$(($per/$Total))

Echo “Free Memory = $per %”

#####FUNCTIONS####################

If you assign a name to set of commands is function

###Syntax##

FNAME() {

commands

commands

commands

}

Call the function #FNAME

Function is a types of commands

Four types of commands

1. Binary files (/bin, /sbin)
2. Shell built in commands (. Command, history)
3. Functions
4. Alias

Every commands check the alias ,then function, shell built then binary files if not it shows commands not found

Function is a semi command

New Process ID will not get for function, alias , and shell built in…

###Properties of function#####

1. Read Write readonly –f funcname

##Example script for func

ADDFUNC() {

Echo “add = $(($a+$b))”

}

##Main program

A=6

B=9

ADDFUNC

1. Local

Export –f funcname

. script

Unset –f funcname

###return command in function example

Func() {

Echo hello

Return

Echo bye

}

Myfunc

Echo exit status is $? (Return command return the status from the code)

Func() {

Echo hello

Return

Echo bye

}

Func

###in the code hello will print after hello bye will not show because we already return the code after hello…

###type command will show what is the type of command###

We can create shell script automatically using function###

####Taking input from user##########

1. While executing

Ssh host

Read command

1. Before executing

Cp cmd ###cp file 1 file 2 <Enter>

Special variables

Getopts

###read command#########

#!/bin/bash

echo “Enter your name:”

read name

echo “Enter your location:”

read loc

echo –e “Your name = $name\nYour location = $loc”

read command option ---- -p and –s option

-p is for print message before reading

Read –p ‘Enter your name’ name

Read –p ‘Enter your loaction’ loc

Read –s –p “enter your passwd’ passwd

###while executing ---- read is useful

Echo –e “\nYour name =$name\nyour location = $loc

-s option in read can be using for secret access like passwd

####SPECIAL VARIABLES#####################

./script 10 20 <Enter>

What are the spl variables

$? Is for exit status

$$ to know pid for script

$! Last executed background job to check through the command

$\* and $@ -- all the values parsed

$# - number of values

$0 - $9 – script name is $0

Cp file 1 file 2

Cp is $0 and file 1 is $1 and file 2 is $2

##For example in cron job script we can use special variables####

Special variables in functions also…

###### case commands#####

Case syntax---

case $var in

patt1) commands;;

patt2) commands;;

pattn) commands;;

esac

###Example script for case###

#!/bin/bash

Read –p ‘Enter val1:’ a

Read –p ‘Enetr val2:’ b

Read –p ‘Enter Operator{ADD|SUB|MUL|DIV}:’ op

Case $op in

ADD) echo “Addition = $(($a+$b))”;;

esac

#### tr command used to convert from lower to upper case #####

Echo hai | tr [:lower:] [:upper:]

###Example script for monitoring####

Using function and case

MENU() {

echo –e “\e[4mMonitoring Menu\e[0m”

echo “1. Check mem”

echo “2. Check cpu”

echo “3. Check disk”

echo “4. Exit”

read –p ‘Select a value> ‘ val

}

##Main Prog

MENU

Case $val in

1. Echo “Total mem = $(free –m | awk ‘/Mem/ {print $2}’)”

Echo “Free mem = $(free –m | awk ‘/Mem/ {print $4}’)”

;;

1. Uptime | awk –F, ‘{print $(NF-2)}’;;
2. Exit;;

\*) MENU;;

Esac

###In the above script we are using function, case and read ######

#####if command#############################################

It overcome from disadvantages of case commands

1. Simple if
2. If-else
3. Else-if

Syntax for simple if

if [ expr ]

then

Commands

Fi

Synatx for if-else

if [ expr ]

then

Commands

else

Commands

Fi

Syntax for else if

if [ expr1 ]

then

Commands1

elif [ expr2 ]

then

Commands2

.

.

.

elif [ exprn ]

then

Commandsn

else

Commands

fi

##checking all the time expressions

1. String comparisons

Operators =, != equal to , not equal to

-z and –n

2. numerical comparisons

-eq, -ne, -gt , -ge, -lt, -le

Usage()

Usage is used to show some error of scripts

3.File test

Seven types of files

Regular (-f), directory, block, character, link(-h), names pipe, socket(-S)…..

File owner and permission

Owner, group owner, read, write, exe, suid, sgid, sticky bit(-k)….

File Size

Empty (! –s)

Non empty (-s)

-a --logical AND

-o – logical OR

###if command syntax

If [ expr ]

if command; then

commands

fi

test command syntax

test command is

[ syntax ] --- directly opens with square brackets

########