13 June 2020

14:38

Shell Scripting

1.Briefing about Echo command

it will print the same info what we mention in double quote

echo "welcome to shell scripting"

Example below

[root@client ~]# echo "welcome to training"

output: welcome to training

2.we can use echo command to display variable

echo "$USER"

3.To execute a command with echo command we can use below command

echo "$(ls -l)"

echo "`ls -l`"

4.syntax of echo command in shell scripting is

echo [-enE] "[arguments]"

when -n is option is used the triling line will be suppressed

[root@naveenhost section1]# cat echon.sh

#!/bin/bash

#This script will talk about -n option in echo

echo "my name is naveen"

echo "iam learning shell"

echo -n "why to learn"

echo "shell"

echo "please use it"

[root@naveenhost section1]# ./echon.sh

my name is naveen

iam learning shell

why to learnshell

please use it

Because of -n cursor will not go to next line and whatever present

in trailing line will be displayed on the same line

5.when the -e option is given then the following backslash-escaped

characters will be interpreted/executed

[root@naveenhost section1]# echo  "s.no\tname\tmarks"

s.no\tname\tmarks

[root@naveenhost section1]# echo -e "s.no\tname\tmarks"

s.no    name    marks

[root@naveenhost ~]# echo "welcome to shellscripting\nwhere is my notebook"

welcome to shellscripting\nwhere is my notebook

[root@naveenhost ~]# echo -e"welcome to shellscripting\nwhere is my notebook"

-ewelcome to shellscripting\nwhere is my notebook

[root@naveenhost ~]# echo -e "welcome to shellscripting\nwhere is my notebook"

welcome to shellscripting

where is my notebook

examples of echo -e command

#\b This is used to remove one letter backside,if i use two times \b\b

it will remove two letters in the backside

[root@naveenhost ~]# echo -e "one\btwo"

ontwo

[root@naveenhost ~]# echo -e "hello\bhi"

hellhi

[root@naveenhost ~]# echo -e "Hello\b\bhilesa"

Helhilesa

#by using \r option it will take second word first and replace

The letters of first word,if the second word is longer than first word

it completely replaces it

[root@naveenhost ~]# echo -e "Hello\rhi"

hillo

[root@naveenhost ~]# echo -e "Hello\rhilesa"

hilesa

#\t is for tab

[root@naveenhost ~]# echo -e "Hello\thilesa"

Hello   hilesa

#\v is for vertical tab

[root@naveenhost ~]# echo -e "Hello\vhilesa"

Hello

     hilesa

#if i want to ignore the backslash then i can use \\ before \v \r \t \b

[root@naveenhost ~]# echo -e "Hello\\\vhilesa"

Hello\vhilesa

#echo -E command is nothing but normal echo command without -e option

colors with echo command and \033[0m is normal clour i mean white

[root@naveenhost ~]# echo -e "\033[0;32mhello"

hello

[root@naveenhost ~]# echo -e "\033[0;32mhello\033[0m"

hello

[root@naveenhost ~]# echo -e "\033[2;32mhello\033[0m"

hello

[root@naveenhost ~]# echo -e "\033[4;32mhello\033[0m"

hello

[root@naveenhost ~]# echo -e "\033[1;32mhello\033[0m"

hello

==================================================================

Variables

==================================================================

=>variables plays an important role in shell scripting

=>variables are used to store data/value

=>Define/declare a variable

ex:-    x=2  (don't provide space on either side of = symbol)

=>use echo $x or echo "$x" or echo "${x}"

these are correct way to use an variable

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

[root@naveenhost section1]# x=2

[root@naveenhost section1]# echo $x

2

[root@naveenhost section1]# echo "$x"

2

[root@naveenhost section1]# echo "${x}"

2

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

#quotations are mandatory when we have space in variable

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

[root@naveenhost section1]# n=shell scripting

bash: scripting: command not found...

[root@naveenhost section1]# n="shell scripting"

[root@naveenhost section1]# echo $n

shell scripting

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

#how to declare a command in an variable,see below

cmd=$(ls -l) or cmd=`date`

use #env command to see system variables

userdefinedvariables are defined by us like x=2 y=5

Note:-The name of an variable can only contain letters(A to Z)

numbers(0-9) and underscore character(\_)

Note:- use env command to list system variables

Note:-Donat start the variable with an number and it should start letter

or an underscore (my\_name)

=================================================================

In this section we are going to write two shell scripts

and accessing then from another script

[root@naveenhost section2]# cat httpd\_start.sh

#!/bin/bash

sudo systemctl start httpd

[root@naveenhost section2]# cat example.sh

#!/bin/bash

#This script will explain to run a script inside a script

/shellscripting/section2/httpd\_start.sh

[root@naveenhost section2]# cat httpd\_stop.sh

#!/bin/bash

sudo systemctl stop httpd

accessing variable in another file in current script

[root@naveenhost section2]# cat one.txt

name="first shell script"

[root@naveenhost section2]# cat shell.sh

#!/bin/bash

source one.txt

echo "$name"

=================================================================

Exit status of a command

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

Each linux command returns a status when it is executed

Exit status is an integer value

If the command is successful we can get output as 0 by using below command

[root@naveenhost section2]# ls

example.sh  httpd\_start.sh  httpd\_stop.sh  one.txt  shell.sh  variable.sh

[root@naveenhost section2]# echo $?

0

[root@naveenhost section2]# llssll

bash: llssll: command not found...

[root@naveenhost section2]# echo $?

127

How to store exit status in a variable

[root@naveenhost section2]# lsss

bash: lsss: command not found...

[root@naveenhost section2]# cmnd\_rc=$?

[root@naveenhost section2]# echo $cmnd\_rc

127

==================Basics of shell===================END===========

=====================Filter command GREP============Start=========

Grep command

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

grep is a filter command, it is used to search anything in a file

#grep above test.txt

#grep lines test.txt

#grep "lines" test.txt

#grep "example script" test.txt

#grep "bash" one.txt two.txt(This will search bash in two files)

#grep "bash" . (this will search in all files in a current directory)

[root@naveenhost section2]# echo "This is a simple info" | grep "simple"

This is a simple info

[root@naveenhost section2]# echo "This is a simple info" | grep "where"

[root@naveenhost section2]#

#cat test.txt | grep "above"

Basic options:-

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

-i -w -v -o -n -c -A -B -C -r -l -h

-i to ignore

[root@naveenhost section2]# echo "This is a simple info" | grep "this"

[root@naveenhost section2]# echo "This is a simple info" | grep -i "this"

This is a simple info

[root@naveenhost section2]#

-w to match a whole word

with this option only exact match is displayed

[root@naveenhost section2]# echo -e "This is a simple info\ninformation" | grep "info"

This is a simple info

information

[root@naveenhost section2]# echo -e "This is a simple info\ninformation" | grep -w "info"

This is a simple info

[root@naveenhost section2]#

-v to display the lines which are not having given string or text

[root@naveenhost section2]# echo -e "This is a simple info\ninformation\nwhere is my bike\nneed to clean it" | grep -v "info"

where is my bike

need to clean it

[root@naveenhost section2]#

-o option to display/print only matched parts from matched lines

[root@naveenhost section2]# echo -e "This is a simple info\ninformation\nwhere is my bike\nneed to clean it" | grep -o "info"

info

info

[root@naveenhost section2]#

-n to display matched line numbers

[root@naveenhost section2]# echo -e "This is a simple info\ninformation\nwhere is my bike\nneed to clean it" | grep -n "info"

1:This is a simple info

2:information

[root@naveenhost section2]#

-c will count the lines that matches the word

[root@naveenhost section2]# echo -e "This is a simple info\ninformation\nwhere is my bike\nneed to clean it" | grep -c "info"

2

[root@naveenhost section2]#

-A to display N lines after match -A 1(after searhed word one line it will display)

[root@naveenhost section2]# echo -e "This is a simple info\ninformation\nwhere is my bike\nneed to clean it" | grep -A 1 "info"

This is a simple info

information

where is my bike

[root@naveenhost section2]# echo -e "This is a simple info\ninformation\nwhere is my bike\nneed to clean it" | grep -A 2 "info"

This is a simple info

information

where is my bike

need to clean it

[root@naveenhost section2]#

-B before match line

[root@naveenhost section2]# echo -e "Hi\nwhere is my requirement\nThis is a simple info\ninformation\nwhere is my bike\nneed to clean it" | grep -B 2 "info"

Hi

where is my requirement

This is a simple info

information

[root@naveenhost section2]#

[root@naveenhost section2]# echo -e "Hi\nwhere is my requirement\nThis is a simple info\ninformation\nwhere is my bike\nneed to clean it" | grep -B 1 "info"

where is my requirement

This is a simple info

information

[root@naveenhost section2]#

-C to display N numbers around match

[root@naveenhost section2]# echo -e "Hi\nwhere is my requirement\nThis is a simple info\ninformation\nwhere is my bike\nneed to clean it" | grep -C 1 "info"

where is my requirement

This is a simple info

information

where is my bike

[root@naveenhost section2]#

[root@naveenhost section2]# echo -e "Hi\nwhere is my requirement\nThis is a simple info\ninformation\nwhere is my bike\nneed to clean it" | grep -C 2 "info"

Hi

where is my requirement

This is a simple info

information

where is my bike

need to clean it

[root@naveenhost section2]#

-r to search under current directory and its sub directory

[root@naveenhost shellscripting]# grep -r "bash" \*

section1/httpd.sh:#!/bin/bash

section2/variable.sh:#!/bin/bash

section2/httpd\_stop.sh:#!/bin/bash

section2/httpd\_start.sh:#!/bin/bash

section2/example.sh:#!/bin/bash

section2/shell.sh:#!/bin/bash

[root@naveenhost shellscripting]#

-l to display only file names

[root@naveenhost section2]# grep -l "bash" \*

example.sh

httpd\_start.sh

httpd\_stop.sh

shell.sh

variable.sh

[root@naveenhost section2]#

-h to hide file names

[root@naveenhost section2]# ls

example.sh  httpd\_start.sh  httpd\_stop.sh  one.txt  shell.sh  variable.sh

[root@naveenhost section2]# grep -h "bash" \*

#!/bin/bash

#!/bin/bash

#!/bin/bash

#!/bin/bash

#!/bin/bash

[root@naveenhost section2]#

Advanced grep command:-

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

Advanced options in grep command are -f -e -E

-f Takes search string/pattern from a file, one per line

[root@naveenhost section3]# cat mysearch

info

[root@naveenhost section3]# echo "information provided is correct\nKindly check with customer" | grep -f mysearch

information provided is correct\nKindly check with customer

[root@naveenhost section3]#

-f option will take strings from a file and used to grep the content in other file

in the above example we put info in mysearch,by using -f and mysearch

we are grepping the other file that is echo content

[root@naveenhost ~]# echo -e "information provided is correct\nKindly check with customer\ncustomer mentioned to hold it for 1 day\nand today customer confirmed\nto close the case" | grep -e "prov" -e "customer" -e "day"

information provided is correct

Kindly check with customer

customer mentioned to hold it for 1 day

and today customer confirmed

[root@naveenhost ~]#

[root@naveenhost section3]# ls

internet  ipv4  mysearch

[root@naveenhost section3]# clear

[root@naveenhost section3]# cat mysearch

info

[root@naveenhost section3]# ls

internet  ipv4  mysearch

[root@naveenhost section3]# grep -f mysearch internet

[root@naveenhost section3]# vim internet

[root@naveenhost section3]# vim internet

[root@naveenhost section3]# vim mysearch

[root@naveenhost section3]# grep -f mysearch internet

Natarajan Chandrasekaran is Chairman of the Board of Tata Sons

The holding company and promoter of more than 100 Tata operating companies

With aggregate annual revenues of more than US $100 billion

He joined the Board of Tata Sons in October 2016.

And was appointed Chairman in January 2017.

ffffffffffffffff

offffffffffff

oaaaaaaaaaaaaaaa

[root@naveenhost section3]# vim mysearch

[root@naveenhost section3]# grep -f mysearch internet

Natarajan Chandrasekaran is Chairman of the Board of Tata Sons

The holding company and promoter of more than 100 Tata operating companies

With aggregate annual revenues of more than US $100 billion

He joined the Board of Tata Sons in October 2016.

And was appointed Chairman in January 2017.

ffffffffffffffff

offffffffffff

oaaaaaaaaaaaaaaa

[root@naveenhost section3]# grep -if mysearch internet

Natarajan Chandrasekaran is Chairman of the Board of Tata Sons

The holding company and promoter of more than 100 Tata operating companies

With aggregate annual revenues of more than US $100 billion

He joined the Board of Tata Sons in October 2016.

And was appointed Chairman in January 2017.

ffffffffffffffff

offffffffffff

oaaaaaaaaaaaaaaa

[root@naveenhost section3]# grep -e "the" -e "shell" -e "of" internet

Natarajan Chandrasekaran is Chairman of the Board of Tata Sons

The holding company and promoter of more than 100 Tata operating companies

With aggregate annual revenues of more than US $100 billion

He joined the Board of Tata Sons in October 2016.

offffffffffff

[root@naveenhost section3]# grep -E "the|shell|of" internet

Natarajan Chandrasekaran is Chairman of the Board of Tata Sons

The holding company and promoter of more than 100 Tata operating companies

With aggregate annual revenues of more than US $100 billion

He joined the Board of Tata Sons in October 2016.

offffffffffff

[root@naveenhost section3]#

-E is used to simplify command to search multiple strings/pattern

in a file

[root@naveenhost ~]# echo -e "information provided is correct\nKindly check with customer\ncustomer mentioned to hold it for 1 day\nand today customer confirmed\nto close the case" | grep -E "for|it|Kindly|check|customer"

information provided is correct

Kindly check with customer

customer mentioned to hold it for 1 day

and today customer confirmed

=>pattern is astring and it represents more than one string

-E To work with patterns

^abc this displays the line starts with abc

abc$ this displays the line ends with abc

grep -E[options] "pattern" file/files

[root@naveenhost section3]# grep -E "^The" internet

The holding company and promoter of more than 100 Tata operating companies

[root@naveenhost section3]#

#grep "^The" internet ==>This command will print the lines which are starting with "The"

[root@naveenhost section3]# grep -E "^Listen" /etc/httpd/conf/httpd.conf

Listen 80

[root@naveenhost section3]#

#grep "2017$" internet

=>By using $ at the end of the string it displays the lines that

end with the searched string

[root@naveenhost section3]# grep -E "2017$" internet

And was appointed Chairman in January 2017

[root@naveenhost section3]#

^$ it display empty lines

[root@naveenhost section3]# grep -E "^$" internet

[root@naveenhost section3]#

\ to remove speacial purpose of any symbol need to use like \^ \$

[root@naveenhost section3]# grep -E "m..e" internet

The holding company and promoter of more than 100 Tata operating companies

With aggregate annual revenues of more than US $100 billion

[root@naveenhost section3]#

. means any character

by using \. it display the lines which have .

[root@naveenhost section3]# grep -E "\." internet

He joined the Board of Tata Sons in October 2016.

And was appointed Chairman in January 2017.

[root@naveenhost section3]#

\b match the empty space at the edge of word

[root@naveenhost section3]# grep -E "Chairman\b" internet

Natarajan Chandrasekaran is Chairman of the Board of Tata Sons

And was appointed Chairman in January 2017.

[root@naveenhost section3]# grep -E "And\b" internet

And was appointed Chairman in January 2017.

[root@naveenhost section3]# grep -E "\bAnd\b" internet

And was appointed Chairman in January 2017.

[root@naveenhost section3]# grep -Ew "And" internet

And was appointed Chairman in January 2017.

[root@naveenhost section3]#

? it means the preceeding character can be 0 times or 1 times

in the below example it prints of and also the word starts with o

[root@naveenhost section3]# grep -E "of?" internet

Natarajan Chandrasekaran is Chairman of the Board of Tata Sons

The holding company and promoter of more than 100 Tata operating companies

With aggregate annual revenues of more than US $100 billion

He joined the Board of Tata Sons in October 2016.

And was appointed Chairman in January 2017.

offffffffffff

oaaaaaaaaaaaaaaa

[root@naveenhost section3]#

\* The preceeding character may be 0 or more times

+ atleast one time of f and maximum any number of times with file

[root@naveenhost section3]# grep -E "of+" internet

Natarajan Chandrasekaran is Chairman of the Board of Tata Sons

The holding company and promoter of more than 100 Tata operating companies

With aggregate annual revenues of more than US $100 billion

He joined the Board of Tata Sons in October 2016.

offffffffffff

[] it will check the letters individually

[root@naveenhost section3]# grep -E "a|t|f|l" internet

Natarajan Chandrasekaran is Chairman of the Board of Tata Sons

The holding company and promoter of more than 100 Tata operating companies

With aggregate annual revenues of more than US $100 billion

He joined the Board of Tata Sons in October 2016.

And was appointed Chairman in January 2017.

ffffffffffffffff

offffffffffff

oaaaaaaaaaaaaaaa

[root@naveenhost section3]# grep -E "[atfl]" internet

Natarajan Chandrasekaran is Chairman of the Board of Tata Sons

The holding company and promoter of more than 100 Tata operating companies

With aggregate annual revenues of more than US $100 billion

He joined the Board of Tata Sons in October 2016.

And was appointed Chairman in January 2017.

ffffffffffffffff

offffffffffff

oaaaaaaaaaaaaaaa

[a-f] it display either a or b or c or d or e or f letters

[root@naveenhost section3]# grep -E "[a-f]" internet

Natarajan Chandrasekaran is Chairman of the Board of Tata Sons

The holding company and promoter of more than 100 Tata operating companies

With aggregate annual revenues of more than US $100 billion

He joined the Board of Tata Sons in October 2016.

And was appointed Chairman in January 2017.

ffffffffffffffff

offffffffffff

oaaaaaaaaaaaaaaa

[a-ds-z] is equal to abcdstuvwxyz

"^[NTH]" it will display the lines starting with N T H

[root@naveenhost section3]# grep -E "^[NTH]" internet

Natarajan Chandrasekaran is Chairman of the Board of Tata Sons

The holding company and promoter of more than 100 Tata operating companies

He joined the Board of Tata Sons in October 2016.

[root@naveenhost section3]#

"[^NTH] it will display the lines other than lines starting N T H

f

[root@naveenhost section3]# grep -E "of{1}" internet

Natarajan Chandrasekaran is Chairman of the Board of Tata Sons

The holding company and promoter of more than 100 Tata operating companies

With aggregate annual revenues of more than US $100 billion

He joined the Board of Tata Sons in October 2016.

offffffffffff

[root@naveenhost section3]# grep -E "of{2}" internet

offffffffffff

[root@naveenhost section3]# grep -E "of{3}" internet

offffffffffff

[root@naveenhost section3]#

lower case uppercase digit

grep -E "[[:lower:]]" internet

grep -E "[[:upper:]]" internet

grep -E "[[:digit:]]" internet

The below command displays only directories because directories start with data/value

ls -ltr | grep -E "^d"

The below command displays only files because files start with -

ls -ltr | grep -E "^-"

# to filter only ipv4,the ipv4 have only 3 or 2 or 1 but not 4

[root@naveenhost section3]# cat ipv4 | grep -E "\b[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\b"

22.32.43.56

222.58.92.66

======================Filter commands GREP===========END==========

==========================Filter commands CUT========Start========

Cut command:-

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

cut -c 1 /etc/passwd

cut -c 1-10 /etc/passwd

cut -c -10 /etc/passwd

cut -c 1- /etc/passwd

cut -c 1-18 /etc/passwd

cut -c 1-18,19-39 /etc/passwd

cut -c 1-18,22 /etc/passwd

cut command based on fields

cut will consider only tab spaces

if there is no tabe space and have only normal space that eneter line will

be consider as an single tab space

#cut -f 2 test.txt

#cut -d ":" -f 5 /etc/passwd

#cut -d ':' -f 1,4 /etc/passwd --output-delimiter=" "

Both the below commands are same -4 or 1-4

#cut -d ':' -f 1-4 /etc/passwd --output-delimiter=" "

#cut -d ':' -f -4 /etc/passwd --output-delimiter=" "

#cut -d ' ' -f 1 file.txt

Here s is used to supress

[root@naveenhost section4]# cut -d ' ' -f 2 file.txt

one     two     three

four    five    six

eight   nine    ten

is

[root@naveenhost section4]# cut -d ' ' -sf 2 file.txt

is

[root@naveenhost section4]# httpd -v | grep "version" | cut -d ':' -f 2 | cut -d '/' -f 2 | cut -d ' ' -f 1

2.4.6

cut -b 2 mytext

cut -b 3,7 mytext

cut -b 1-7 mytext

cut -b 1-7,8-19 mytext

cut -b 5- mytext

cut -b -10 mytext

cut -b -7,9 mytext

use --complement to complement the ouput it means it will display other than given

#cut --complement 2 mytext (it will display other than 2)

======================Filter commands CUT============END==========

=====================Filter commands AWK===============Start======

use cut command to filter characters

use awk command to filter fields

#httpd -v | awk -F '[ /]' '/version/{print $4}'

#httpd -v | awk -F '[ /]' 'NR==1 {print$4}'

2.4.6

The awk command is a powerful method for processing or analyzing

text or data files,which are organized by lines(rows or records) and

columns(fields)

we can use awk as a linux command and also as a scripting language

like bash shell scripting

In awk the tab and space is default seperator

awk [options] '[selection\_critetia]{action}' file

-F to specify a field seperator (default seperator is tab and space)

-f file to specify a file that contains awk script

-v var=value to declare a variable

selection criteria: pattern/condition

action is a logic to perform to action each row/column

[root@naveenhost section4]# awk '{print $1}' file.txt

one

four

eight

where

$0 and {print} both are same and print whole output

[root@naveenhost section4]# awk '{print $0}' file.txt

one     two     three

four    five    six

eight   nine    ten

where is my bike please take and come

[root@naveenhost section4]# awk '{print}' file.txt

one     two     three

four    five    six

eight   nine    ten

where is my bike please take and come

output filed seperator OFS

[root@naveenhost section4]# awk 'BEGIN {OFS="\_"} {print $1,$2}' file.txt

one\_two

four\_five

eight\_nine

where\_is

[root@naveenhost section4]#

#awk '{print $3,$1}' file.txt (reverse the columns)

NR will print line or record number

# awk '{print NR}' file.txt

NF will print the maximum columns and it also refer as last column

[root@naveenhost section4]# awk '{print NR,$0,NF}' file.txt

1 one   two     three 3

2 four  five    six 3

3 eight nine    ten 3

4 where is my bike please take and come 8

NF means number of filelds,$NF means it prints last column

[root@naveenhost section4]# awk '{print $NF}' file.txt

three

six

ten

come

NR means line number and if we put $NR it prints first row first field

second row second field third row third field like this till end of file

[root@naveenhost section4]# awk '{print $NR}' file.txt

one

five

ten

bike

=====================Filter commands AWK============END===========

=====================Operators on Strings===========Start=========

Operations on strings:-

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

x=2

echo "$x"      =>one wany of defining variable

echo "${x}"    =>another way of defining variable

echo "$#x}"    =>it displays the length of the variable(finding the string length)

[root@naveenhost section5]# x=chilakavenkatasainaveen

[root@naveenhost section5]# echo $x

chilakavenkatasainaveen

[root@naveenhost section5]# echo "$x"

chilakavenkatasainaveen

[root@naveenhost section5]# echo "${x}"

chilakavenkatasainaveen

[root@naveenhost section5]# echo "${#x}"

23

By using << symbol we can skip all the code in script

#!/bin/bash

<< skip

x=1456

y=${#x}

echo "The value of x is: $x"

echo "The value of x is: ${#x}"

echo "The value of y is: $y"

skip

$(( $x + $y )) concatnation of two values

${name^^} it will make the name variable into uppercase letters

${name,,} it will make the name variable into lowercase letters

name="ShEll"

echo "The uppercase info is: ${name^^}"

echo "The lowercase info is: ${name,,}"

usage of direname and basename on strings/paths

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

dirname   : strip/removes last component from file name

[root@naveenhost section5]# dirname /etc/httpd/conf/httpd.conf

/etc/httpd/conf

[root@naveenhost section5]# basename /etc/httpd/conf/httpd.conf

httpd.conf

How we can substitute this in a variable,see below

[root@naveenhost section5]# httpd\_conf\_path="/etc/httpd/conf/httpd.conf"

[root@naveenhost section5]# dirname $httpd\_conf\_path

/etc/httpd/conf

[root@naveenhost section5]# basename $httpd\_conf\_path

httpd.conf

[root@naveenhost section5]# cat string\_slice.sh

#!/bin/bash

name="shell scripting"

echo "${name}"

echo "${name:0}"

echo "${name:4}"

echo "${name:2:1}"

echo "${name:2:3}"

[root@naveenhost section5]# ./string\_slice.sh

shell scripting   =>print variable

shell scripting   =>print from 0 to end

l scripting       =>print from 4 to end

e                  =>print from 2 and only 1 character,that is why it is e

ell                  =>print from 2 and only 3 characters after 2

Script written in /sheelscripting/section6/tomcat.sh

======================Operators on strings=======END=============

=======================I/O commands===========Start==============

Input and output commands for shell script:-

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

Important point about read

if i didn't mention the variable name for suppose in read -parts

read -p "Enter your name: "

by default it will sstore in $REPLY

script written in /shellscripting/section7/inout.sh

command line arguments/positional arguments:-

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

command line aruguments are the arguments/values specified at

the command prompt while running commands/shellscripts

$0 will display the scrip name

$# will display the number of arguments passed

$@ will display the variables which we enetered

$\* with IFS="," will display the variables which we entered with sperator ,

Note:-if we are printing double digit variable like echo "$10" for suppose

it wil print like $1 and0

for suppose

$1 is shell

it will print like shell0

so to avoid this we need define the varibale in {} like echo "${10}"

=====================I/O commands=========END===================

=====================Arithmetci operators=============Start=======

shell script variables are by default treated as strings,not numbers

which adds some complexity to doing math in shell script

There are different ways to perform arithmetic operations

=>Using declare

=>using expr

=>using let

=>using (( ))

[root@naveenhost section7]# x=7

[root@naveenhost section7]# y=8

[root@naveenhost section7]# ((x+y))

[root@naveenhost section7]# ((sum=x+y))

[root@naveenhost section7]# echo $sum

15

[root@naveenhost section7]# ((sub=x-y))

[root@naveenhost section7]# echo $sub

-1

[root@naveenhost section7]# ((mul=x\*y))

[root@naveenhost section7]# echo $mul

56

[root@naveenhost section7]# ((div=x/y))

[root@naveenhost section7]# echo $div

0

[root@naveenhost section7]# ((rem=x/y))

[root@naveenhost section7]# echo $rem

0

[root@naveenhost section7]# ((x++))

[root@naveenhost section7]# echo $x

8

[root@naveenhost section7]# ((y++))

[root@naveenhost section7]# echo $y

9

[root@naveenhost section7]# ((x--))

[root@naveenhost section7]# echo $x

7

[root@naveenhost section7]# ((y--))

[root@naveenhost section7]# echo $y

8

[root@naveenhost section7]# ((sum=2+10))

[root@naveenhost section7]# echo $sum

12

[root@naveenhost section7]# ((sub=2-10))

[root@naveenhost section7]# echo $sub

-8

[root@naveenhost section7]# ((mul=2\*10))

[root@naveenhost section7]# echo $mul

20

[root@naveenhost section7]# echo "$((4+3))"  ($ is mandatory)

7

[root@naveenhost section7]# echo "((4+3))"

((4+3))

[root@naveenhost section7]# x=4.5

[root@naveenhost section7]# y=3.5

[root@naveenhost section7]# ((sum=x+y))

-bash: ((: 4.5: syntax error: invalid arithmetic operator (error token is ".5")

[root@naveenhost section7]# ((sum=${x}+${y}))

-bash: ((: sum=4.5+3.5: syntax error: invalid arithmetic operator (error token is ".5+3.5")

[root@naveenhost section7]# bc<<<"$x+$y"

8.0

[root@naveenhost section7]# x=245

[root@naveenhost section7]# y=2

[root@naveenhost section7]# bc<<<"scale=2;$x/$y"

122.50

[root@naveenhost section7]# bc<<<"scale=1;$x/$y"

122.5

case Statement:-

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

check the script present in /shellscripting/section8/case.sh

=====================Arithmetic operators====END=================

=====================test command===============Start============

it is a command to judge conditions

simple syntax

test condition or [ condition ] or [[ condition ]]

how to make condition to work with test command

comparision operators

file test operators

[[ int1 -eq int2 ]]

[[ int1 -ne int2 ]]

[[ int1 -gt int2 ]]

[[ int1 -lt int2 ]]

[[ int1 -ge int2 ]]

[[ int1 -le int2 ]]

[[ ! int1 -eq int2 ]]          it reverse the result

[[ -z str ]] it will check the string is empty or not

[[ -n str ]] it return true if length of the string non-zero else false

[[ str1 == str2 ]]

[[ str1 != str2 ]]

File test operators with test command:-

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

[[ -d file ]] it return the file/path is directory else false

[[ -f file ]] it return the file/path is a file else false

[[ -e file ]] it return the file/path is exists else false

[[ -r file ]] it return the file/path is readable else false

[[ -w file ]] it return the file/path is writable else false

[[ -x file ]] it return the file/path is executable else false

command chaining operators:-

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

combining two or more commands is called command chaining

There are different operators to combine commands in shellscripting

semicolon ; operator   => date ; uptime ; ls -ltr ; pwd ; cat /etc/passwd

logical AND && operator=>whenever we are using logical and operator,if first command is successful then second command will execute else it will not

logical OR || operator => command2 is executed if command1 is failed

command 1 && command 2 || command 3

if cmd1 and cmd2 is successful then cmd3 will not executed

if any of the cmd1 or cmd2 is failed then cmd3 is executed

cmd1 || cmd2 && cmd3

if cmd1 if failed then cmd2 and cmd3 will be executed

if cmd1 is successful cmd2 and cmd3 will not executed

if and ifelse scripts are present in =>/shellscripting/section9

httpd script is in => /shellscripting/section9/httpd.sh

Logic to handle command line arguments:-

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

script written on httpd automation and it is in

/shellscripting/section9/

-rwxr-xr-x. 1 root root 858 May  5 08:05 automation\_httpd.sh

-rwxr-xr-x. 1 root root 968 May  5 08:13 en\_automation\_httpd.sh

-rwxr-xr-x. 1 root root 913 May  5 09:01 httpd\_automation\_case.sh

backup script is written chekc this

[root@naveenhost section9]# cat backup.sh

#!/bin/bash

#Author:Naveen

src="/var/log/httpd"

dest="/backups"

date=$(date '+%b-%d-%y-%H-%M-%S')

[[ -e $dest ]] || mkdir $dest

tar -cvpzf $dest/mybackup-${date}.tar.gz $src

-rwxr-xr-x. 1 root root 176 May  5 11:17 backup.sh

=================test command================END===============

===========scheduling jobs with at and cron========Start=======

#at 01:57

[root@naveenhost section9]# at 11:32

at> bash backup.sh

at> <EOT>                        =>This is CTRL+D

job 2 at Tue May  5 11:32:00 2020

[root@naveenhost section9]# at 02:00 AM

at> bash backup.sh

at> <EOT>

job 3 at Wed May  6 02:00:00 2020

[root@naveenhost section9]# echo "bash backup.sh" | at 09:00 PM

job 4 at Tue May  5 21:00:00 2020

[root@naveenhost section9]# atq

3       Wed May  6 02:00:00 2020 a root

4       Tue May  5 21:00:00 2020 a root

[root@naveenhost section9]#

#atq

#atrm 1

#at 10:00 AM

#at 10:00 AM Sun

#at 10:00 AM May 25

#at 10:00 AM 6/22/2020

#at 10:00 AM 6.22.2020

#at 10:00 AM next month

#at 10:00 AM tomorrow

#at now + 1 hour

#at now + 30 minutes

#at now + 1 week

#at now + 2 weeks

#at now + 1 year

#at now + 2 years

#at midnight

The crontab is used for running specific tasks on a regular basis

each user can schedule cronjobs

syntax: minutes(0-59) hours(0-23) dayofthemonth(1-31) month(1-12) dayoftheweek(0-6)

each scheduled job has six fields

#crontab -e

#crontab -eu username

#crontab -l

#crontab -lu username

58 6 30 3 3  /sheelscripting/section9/backup.sh

00 \*/2 \* \* \*  /sheelscripting/section9/backup.sh (every 2 hours)

00 9,21 \* \* \* (script will execute at 9am and 9pm)

00 17 \* \* 0 (every sunday)

00 5 \* \* 0 (every sunday 5am)

0 0 1 1 \* (yearly once means january 1)

@yearly

@monthly

@daily

@hourly

@reboot it is useful for those tasks which you want to run on system

Basic for loop:-

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

for i in list\_of\_values

do

    command1

    command2

done

for i in 1 2 3

do

    echo "This is value"

    echo "The value is $i"

done

for i in $(ls)

do

    echo "This is File Name"

    echo "The File Name Is: $i"

done

check for /shellscripting/section11/loop.sh

C language for loop

for ((i=1;i<=10;i++))

for ((i=1;i<=10;i--))

for ((initialization;condition;increment/decrement))

do

    command1

    command2

done

check for /shellscripting/section11/forcloop.sh

Infinity forloop:-

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

for ((;;))

do

    command1

    command2

done

check for /shellscripting/section11/infinityloop.sh

Shell script to isnatll multiple packages:-

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

mulinstall.sh

handle\_cmd\_line\_arguments.sh

Difference between $@ and $\*,these two represents all the commandline arguments

check section 11 dollar.sh

Loop Control statements or commands:-

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

Simply break and continue are called loop control statements

break and continue commands are used to control the execution of loops

break command is used to terminate/exit current loop completely

before the actual ending of loop

=================Loop=============================END=======

============Remote server===================Start===========

we can connect to remote server using ssh from out local/working server

There are two ways to connect with remote server using ssh

=>Using password or password authentication

=>using ssh-keys(passwordless authentication)

#ssh -t -o StrictHostKeyChecking=No root@192.168.43.206 "top"

sshpass:-

-------

sshpass is a utility/command using that we can provide a password in an non interactive way

#ssh -t -o StrictHostkeyChecking=No root@192.168.43.206 "ls;date;uptime;free -m"

#sshpass -p "Krish@123" ssh -t -o StrictHostkeyChecking=No root@192.168.43.206 "ls;date;uptime;free -m"

#sshpass -p "Krish@123" ssh -t -o StrictHostkeyChecking=No root@192.168.43.206 "ls"

#sshpass -f pass ssh -t -o StrictHostkeyChecking=No root@192.168.43.206 "ls"

In the above command pass is an file and in the password is stored

=>There is an varibale called SSHPASS

#export SSHPASS="Krish@123"

#sshpass -e  ssh -t -o StrictHostkeyChecking=No root@192.168.43.206 "ls"

check for multiple scripts in the same script on ssh

/shellscripting/section12/sshscript.sh

shellscript to execute different commands on different servers with different users and different passwords:-

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

check section12

IFS is an internal field seperator ,it is an shell or environmental veriable

check section13

=================Functions=============Start===========

Functions:-

---------

A function is a block of code that perform a specific task

Functions concept reduces the code length

function function\_name

{

    commands/statements

}

function\_name()

{

    commands/statements

}

First define the function later call the function=>basic rule

There are Two Types of Variables

They are

1.Global

2.Local

Note:-

All variables are global by default

if a variable is defined inside a function or outside a function and able to access is called global variable

local p=25

but local variables should be mentioned inside a fucntion

Return a variable value by using return

return works only for the values but not for strings

Passing paramters to a function:-

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

check the script and notes in /shellscripting/section14/functionimp.sh

addition()

{

    m=$1

    n=$2

        result=$(( $m + $n ))

        echo "The addition of two $m and $n numbers: $result"

}

x=225

y=225

addition $x $y

p=23

q=25

addition $p $y

===============Functions=================END======

Printf:-

------

Both echo and printf commands are used to display String or value of an variable

The basic difference between echo and printf is echo sends a newline at the end of its output

but There is no way to send an EOF in printf command

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

for example see below:-

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

[root@naveenhost section14]# echo "This is echo"

This is echo

[root@naveenhost section14]# printf "This is printf"

This is printf[root@naveenhost section14]#

example agian:-

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

[root@naveenhost section15]# x=20

[root@naveenhost section15]# y=25

[root@naveenhost section15]# z="shell scripting"

[root@naveenhost section15]# printf "%d %d %s\n" "$x" "$y" "$z"

20 25 shell scripting

[root@naveenhost section15]#

==========AWK===================================Start===========

=>The awk command is programming language,which requires no compiling, and allows

the user to use variables,numeric functions, string functions and logical operators

=>The awk command in unix is just like a scripting language which is used for text processing

AWk is used like as a command or as a scripting language

=>awk command in unix is most important command used to find or replace text

HOW awk command works:-

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

=>AWk reads data from a file or from its standard input, and outputs to its standard output

=>AWk views a text file as records and fields

=>Each line is a record and columns in lines/record are called fields

=>By default fields are seperated based on space.(Note: we can also change the field seperator with -F option)

=>AWK command works on each line individually

=>AWk has its won predefined variables like $0 $1 $2 $3.........and NR and NF

-F fs to specify a field seperator

-v var=value to declare a variable

-f file to specify a file that contains awk script

AWk command with action and basic variables:-

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

#cat /etc/passwd | awk -F '[:]' '{print "ok"}'

#cat /etc/passwd | awk -F '[:]' '{print "ok","AWK command"}'

#cat /etc/passwd | awk -F '[:]' '{print "ok" "AWK command"}'

#cat /etc/passwd | awk -F '[:]' '{print "ok""AWK command"}'

Basic variables are $0 $1 $2 $3 $4 $5 $6 $7

$0 means entire file/line

NR->no.of records

NF->

[root@naveenhost section17]# cat demo1.txt | awk '{print $1}'

This

This

This

[root@naveenhost section17]# cat demo1.txt | awk '{print $2}'

is

is

is

[root@naveenhost section17]# cat demo1.txt | awk '{print $1,$3}'

This first

This second

This third

[root@naveenhost section17]# cat demo1.txt | awk '{print $1,$4}'

This line

This line

This line

[root@naveenhost section17]# cat demo1.txt

This is first line

This is second line

This is third line

[root@naveenhost section17]# cat demo1.txt | awk '{print $1,$4,NR}'

This line 1

This line 2

This line 3

[root@naveenhost section17]# cat demo1.txt | awk '{print NR,$1,$4}'

1 This line

2 This line

3 This line

[root@naveenhost section17]# cat demo1.txt | awk '{print NR,$1,$NF}'

1 This line

2 This line

3 This line

[root@naveenhost section17]# cat demo1.txt | awk '{print NR,$1,NF}'

1 This 4

2 This 4

3 This 4

[root@naveenhost section17]# cat demo1.txt | awk '{print NR,$1,$NF}'

1 This line

2 This line

2 This line

3 This line

[root@naveenhost section17]# awk '{print FILENAME }' demo1.txt    demo1.txt

demo1.txt

demo1.txt

[root@naveenhost section17]#

#cat /etc/passwd | awk -F '[:]' 'NR <=3 {print}'

#cat /etc/passwd | awk -F '[:]' 'NR >=3 {print}'

#cat /etc/passwd | awk -F '[:]' '/root/ {print}'

#cat /etc/passwd | awk -F '[:]' '{print $1}'

#cat /etc/passwd | awk -F '[:]' '/root/ {print $1}'

#awk -f awk\_script.awk filename

awk scripting:-

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

Here are the actions

=>BEGIN block is performed before reading the file

=>END block is performed after processing the file

=>Rest of the actions are perofrmed while processing the file

Note:-

----

Need atleast one action to run awk script

No Need of input for BEGIN action

Input is required only for middle actions and END actions

#awk 'BEGIN { print "===working on /etc/passwd file====="} /root/ {print $0} END {print "======completed working on /etc/passwd filen======="}' /etc/passwd

===working on /etc/passwd file=====

root:x:0:0:root:/root:/bin/bash

operator:x:11:0:operator:/root:/sbin/nologin

======completed working on /etc/passwd filen=======

[root@naveenhost section17]# cat myawk.awk

BEGIN  {

print "=======working on /etc/passwd file"

}

/root/ {

print $0}

END {

print "========completed working on etc/passwd file=========="

}

[root@naveenhost section17]# awk -f myawk.awk /etc/passwd

=======working on /etc/passwd file

root:x:0:0:root:/root:/bin/bash

operator:x:11:0:operator:/root:/sbin/nologin

========completed working on etc/passwd file==========

[root@naveenhost section17]# awk 'BEGIN {print "ok"}'

ok

[root@naveenhost section17]# cat awkscript.awk

#!/bin/awk -f

BEGIN {

print "ok"

}

[root@naveenhost section17]# ./awkscript.awk

ok

[root@naveenhost section17]# cat myawk1.awk

BEGIN {

print "ok"

}

[root@naveenhost section17]# awk -f myawk1.awk

ok

[root@naveenhost section17]#

BEGIN is called to initialize variables

[root@naveenhost section17]# awk 'BEGIN { a = 5 ; print a}' (we need to seperate with ; otherwise it will not give output

5

[root@naveenhost section17]# cat simple.awk

BEGIN { a=5 ; print a}

[root@naveenhost section17]# awk -f simple.awk

5

[root@naveenhost section17]#

awk variables and How to read variable values from command line:-

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

The below syntax is for command line

awk options 'BEGIN { } pattern/condition { } END { }' filename

command | awk options 'BEGIN { } pattern/condition { } END { }'

The below syntx is for script

awk 'BEGIN { } pattern/condition { } END { }' filename

command | awk 'BEGIN { } pattern/condition { } END { }'

END { } BEGIN { } are optional but pattern/condition mandatory

[root@naveenhost section17]# echo '2 6'| awk '{print "a=" $1, "b=" $2}'

a=2 b=6

[root@naveenhost section17]#

[root@naveenhost section17]# cat ab.txt

2 6

[root@naveenhost section17]# cat ab.txt| awk '{print "a=" $1, "b=" $2}'

a=2 b=6

[root@naveenhost section17]#

[root@naveenhost section17]# echo '2 6'| awk '{ a=$1; b=$2; print a,b }'

2 6

[root@naveenhost section17]# cat awk\_script.sh

BEGIN {

}

{

a=$1

b=$2

print "a="a, "b="b

}

[root@naveenhost section17]# cat ab.txt

2 6

[root@naveenhost section17]# awk -f awk\_script.sh ab.txt

a=2 b=6

[root@naveenhost section17]# cat awk\_script.awk

#!/bin/awk -f

BEGIN {

}

{

a=$1

b=$2

print "a="a, "b="b

}

[root@naveenhost section17]# cat ab.txt

2 6

[root@naveenhost section17]# ./awk\_script.awk ab.txt

a=2 b=6

[root@naveenhost section17]#

Section1:-

#httpd -v | grep -i "version" | awk -F '[/]' '{print $2}' | awk '{print $1}'

#systemctl status httpd | awk -F '[: ]' 'NR==3 {print $4}'

#cat /etc/httpd/conf/httpd.conf | grep '^Listen' | awk '{print $2}'

#which $SHELL

#whatis pwd

Redirection operators stdin stdout stderr:-

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

- Linux commands needs some input (file or any another attribute) and it results some output

- By default,input is being given with the keyboard,and output/error are displaying on your screen

- sometimes we want to put output of a command into a file,or you may want to issue another command on the output of one command

- In another case,we may want a file to be the input for a command

so we have

- output redirection operator

- input redirection operator

- combining redirections operator

output redirection operator:-

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

>

>>

Input redirection operator:-

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

<   #cat < demo.txt

Combining redirections operator:-

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

| pipe

0 - stdin

1 - stdout

2 - stderr

ls -ltr > demo.txt         ls -ltr 1> demo.txt

ls -mmmdncd 2> demo.txt

[root@naveenhost practice]# ls 1> succ.txt 2>err.txt

[root@naveenhost practice]# cat succ.txt

demo.txt

err.txt

httpd.sh

succ.txt

[root@naveenhost practice]# cat err.txt

[root@naveenhost practice]#

[root@naveenhost practice]# hdfkfj 1>succ.txt 2>err.txt

[root@naveenhost practice]# cat succ.txt

[root@naveenhost practice]# cat err.txt

bash: hdfkfj: command not found...

[root@naveenhost practice]#

In the below example java version output is displaying error place so we use 2>

[root@naveenhost practice]# java -version

openjdk version "1.8.0\_242"

OpenJDK Runtime Environment (build 1.8.0\_242-b08)

OpenJDK 64-Bit Server VM (build 25.242-b08, mixed mode)

[root@naveenhost practice]# java -version 1>succ.txt

openjdk version "1.8.0\_242"

OpenJDK Runtime Environment (build 1.8.0\_242-b08)

OpenJDK 64-Bit Server VM (build 25.242-b08, mixed mode)

[root@naveenhost practice]# cat succ.txt

[root@naveenhost practice]# java -version 2>err.txt

[root@naveenhost practice]# cat err.txt

openjdk version "1.8.0\_242"

OpenJDK Runtime Environment (build 1.8.0\_242-b08)

OpenJDK 64-Bit Server VM (build 25.242-b08, mixed mode)

Below is to store error and success output in same file

[root@naveenhost practice]# java -version 1>succ.txt 2>succ.txt

[root@naveenhost practice]# cat succ.txt

openjdk version "1.8.0\_242"

OpenJDK Runtime Environment (build 1.8.0\_242-b08)

OpenJDK 64-Bit Server VM (build 25.242-b08, mixed mode)

[root@naveenhost practice]#

2>&1 store the error ouput where my successouput is storing

&> store error and success output

[root@naveenhost practice]# java -version 1>succ.txt 2>&1

[root@naveenhost practice]# java -version &>succ.txt

[root@naveenhost practice]#

1>  success ouput

2> error ouput

1> succ.txt 2> succ.txt it stores error and success in same file

1> succ.txt 2>&1 it stores error and success in same file

&> filename it stores error and success in same file

echo:-

----

when we try to display a variable always use double quotes

if suppose i use single quote then it will display what we written in single quote

[root@naveenhost practice]# echo '$user'

$user

[root@naveenhost practice]# echo "welcome to shellscripting"

welcome to shellscripting

[root@naveenhost practice]# user="shell scripting"

[root@naveenhost practice]# echo $user

shell scripting

[root@naveenhost practice]# echo '$user'

$user

[root@naveenhost practice]# echo "$user"

shell scripting

[root@naveenhost practice]# echo "${user}"

shell scripting

[root@naveenhost practice]# echo $user

shell scripting

[root@naveenhost practice]#

===============================================

To execute a command with echo command then use

[root@naveenhost practice]# echo $(uptime)

17:06:54 up 3:26, 3 users, load average: 0.01, 0.03, 0.05

[root@naveenhost practice]# echo "$(uptime)"

17:07:00 up  3:26,  3 users,  load average: 0.01, 0.03, 0.05

[root@naveenhost practice]# echo '$(uptime)'

$(uptime)

[root@naveenhost practice]# echo `$(uptime)`

bash: 17:07:43: command not found...

[root@naveenhost practice]# echo `uptime`

17:07:51 up 3:27, 3 users, load average: 0.04, 0.04, 0.05

[root@naveenhost practice]# echo "`ls -l`"

total 0

[root@naveenhost practice]#

================================================

[root@naveenhost practice]# cat echon.sh

#!/bin/bash

echo "Welcome to shell scripting"

echo "where is my book"

echo -e "today is the day to study"

echo -n "this is my shell"

echo "scripting"

echo "learn as you can"

[root@naveenhost practice]# ./echon.sh

Welcome to shell scripting

where is my book

today is the day to study

this is my shellscripting

learn as you can

[root@naveenhost practice]#

====================================================

\b removes one ltter in backside

\b\b removes two letters in backsie

\r carriage return,it will replace he with hi

[root@naveenhost practice]# echo -e "hello\rhi"

hillo

it will replace three letters hel will replace with and

[root@naveenhost practice]# echo -e "hello\rand"

andlo

[root@naveenhost practice]#

\v vertical tab

\t for tab

\c will compress next output

[root@naveenhost practice]# echo -e "hello\cwhere are you"

hello[root@naveenhost practice]#

\a will give sound

\\ eliminate the special purpose

[root@naveenhost practice]# echo -e "hello\\\vwhere are you"

hello\vwhere are you

[root@naveenhost practice]#

-E if i use -E with -e then it is like normal echo command

[root@naveenhost practice]# echo -eE "hi\nhello"

hi\nhello

accessing variables from othe script and other file:-

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

source ./one.sh

then i can use variables present in that shell script one.sh

source one.txt

i can use variables from one.txt

Storing exit status value in variable:-

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

[root@naveenhost practice]# lssss

bash: lssss: command not found...

[root@naveenhost practice]# exit\_status=$?

[root@naveenhost practice]# echo $exit\_status

127

[root@naveenhost practice]#