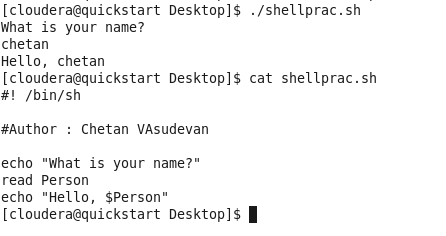
Shell Scripting Basics to start with

Shell is a command line Interpreter

To script/automate some tasks or sequential tasks we use shell scripting

Takes i/p from user -> translates commands -> language understood by kernel



Above is the sample shell script you have written, now change the mode before you execute

Chmod 777 shellprac.sh

After that execute it by ./shellprac.sh

The script will execute and ask your name and print it appended with Hello.

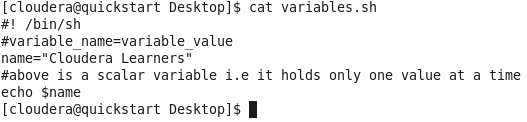
Using variables : it’s a pointer to the actual data, it’s a character string to which we assign values.

Local variables: is a variable that is present within the current instance of the shell. Not available to programs that are started by the shell.

Environment variables: is a variable available to any child process of the shell. Some program need environment variable in order to function properly

Shell variable: like global variables there are shell variables and is set by the shell and is required by the shell in order to function properly.

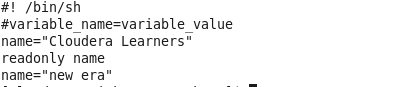
Note: shell scripting has no space between variable assignment and declaration ex: varname=varvalue



Change the mode chmod 777 variables.sh

./variables.sh – will result in Cloudera Learners

Read-only variable, if you put read only in front of any variable, you cannot change or assign new value to that variable.



After changing mode and executing it – it will return in ./fiename.sh: line 5: name: read-only variable

Unset variables – cleans the values assigned to that variable

[cloudera@quickstart Desktop]$ cat variables\_unset1.sh

#! /bin/sh

#variable\_name=variable\_value

name="Cloudera Learners"

unset name

after changing mode and running the above script it will result in - nothing because we have unset the variables which we have already set

special variables:

$0 – filename of the script

$1 . . . .$9 – arguments passing to the script as you wish to

$# - return no of arguments passed to the script

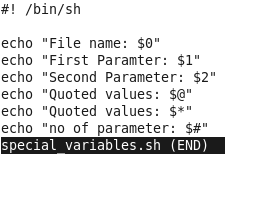
$\* - returns the number of arguments passed to the script that are double quoted

$@ - returns all arguments that are individually double quoted to the script

$? – exit status of the last command you have executed

$$ - process number of current shell also process id

Lets see how they behave, below is the script I wrote and name it as special\_variable.sh



Run1: - ./special\_variables.sh cloudera learner

Below is the output

File name: ./special\_variables.sh

First Parameter: cloudera

Second Parameter: learner

Quoted values: cloudera learner

Quoted values: cloudera learner

no of parameter: 2

Run2:- ./special\_variables.sh "cloudera learner"

Below is the output

File name: ./special\_variables.sh

First Paramter: cloudera learner

Second Parameter:

Quoted values: cloudera learner

Quoted values: cloudera learner

no of parameter: 1

Run3:- ./special\_variables.sh "cloudera" "learner"

Below is the output

File name: ./special\_variables.sh

First Paramter: cloudera

Second Parameter: learner

Quoted values: cloudera learner

Quoted values: cloudera learner

no of parameter: 2

Usage of $TOKEN – a special variable (below is sample code)



Now change the mode and run this script by ./sample\_token.sh cloudera is best among the other distributions

Below is the output (it is like a for each loop)

cloudera

is

best

among

the

other

distributions

echo $? – gives the exit status of the previous command we ran, if exit status is 0 then its successful and if exit status is 1 then its not successful.

Basic Operators in shell scripting

Arithmetic operators

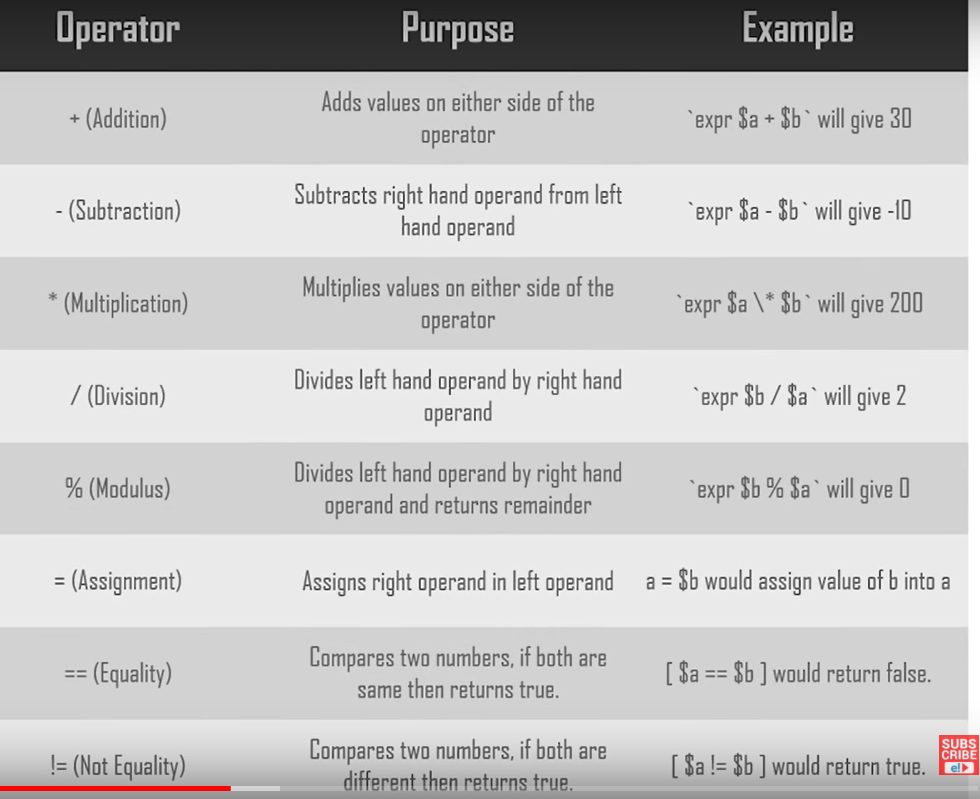
Relational Operators

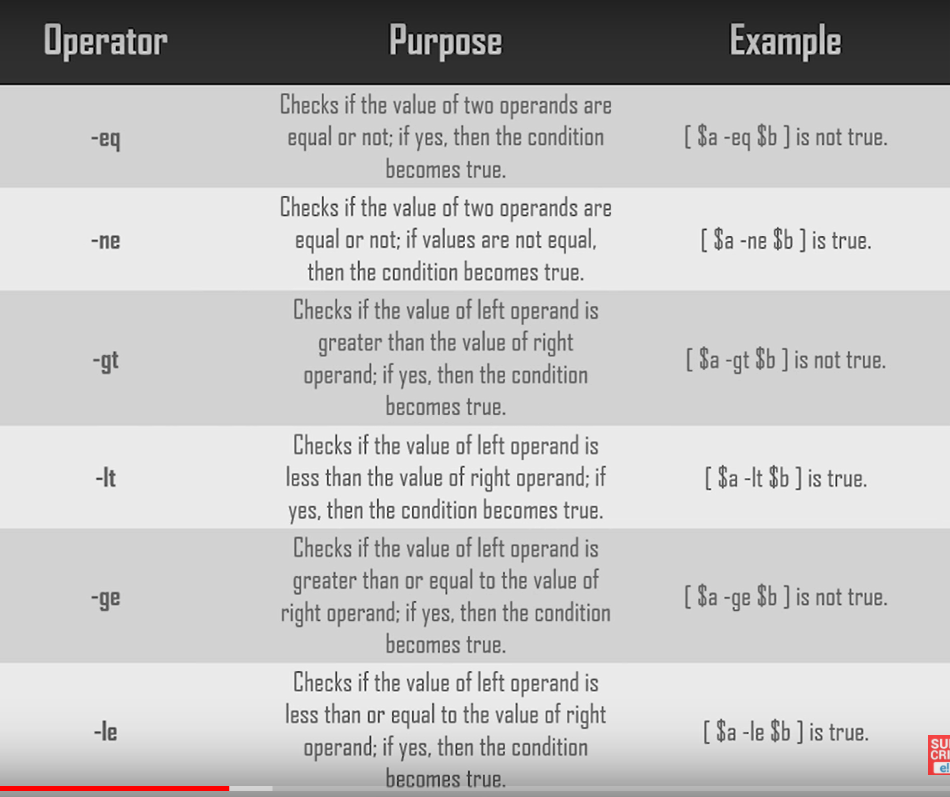
Boolean Operators

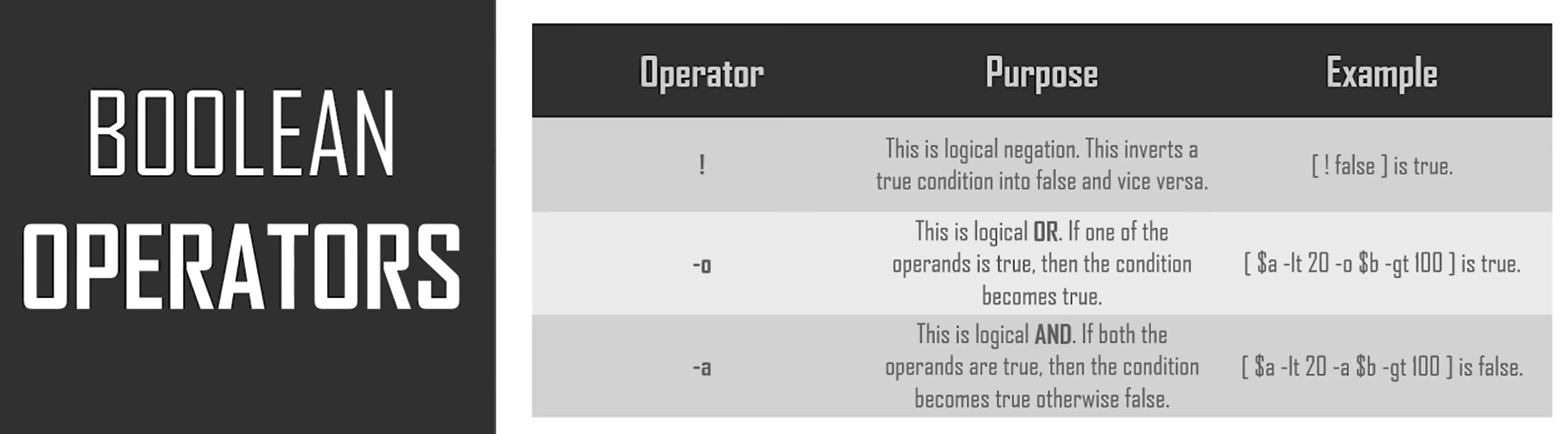
String Operators

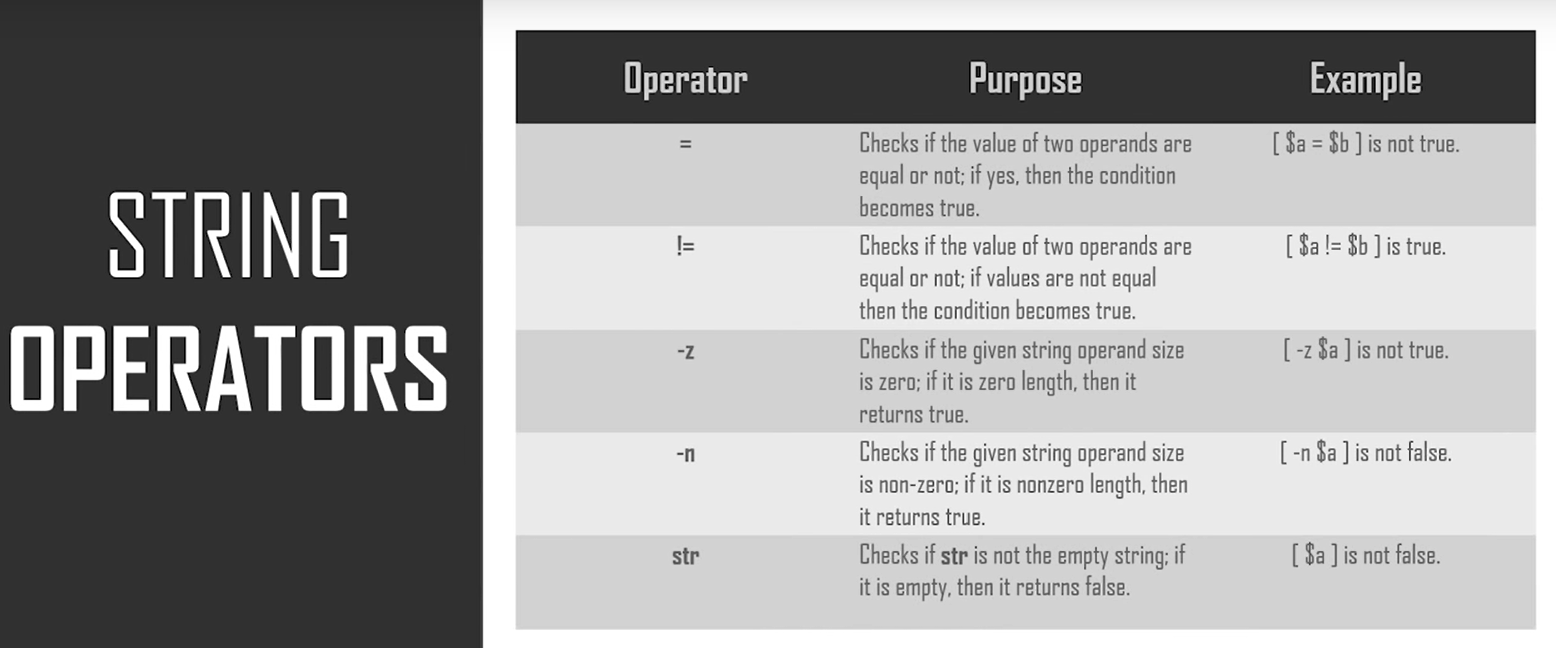
File Test Operators

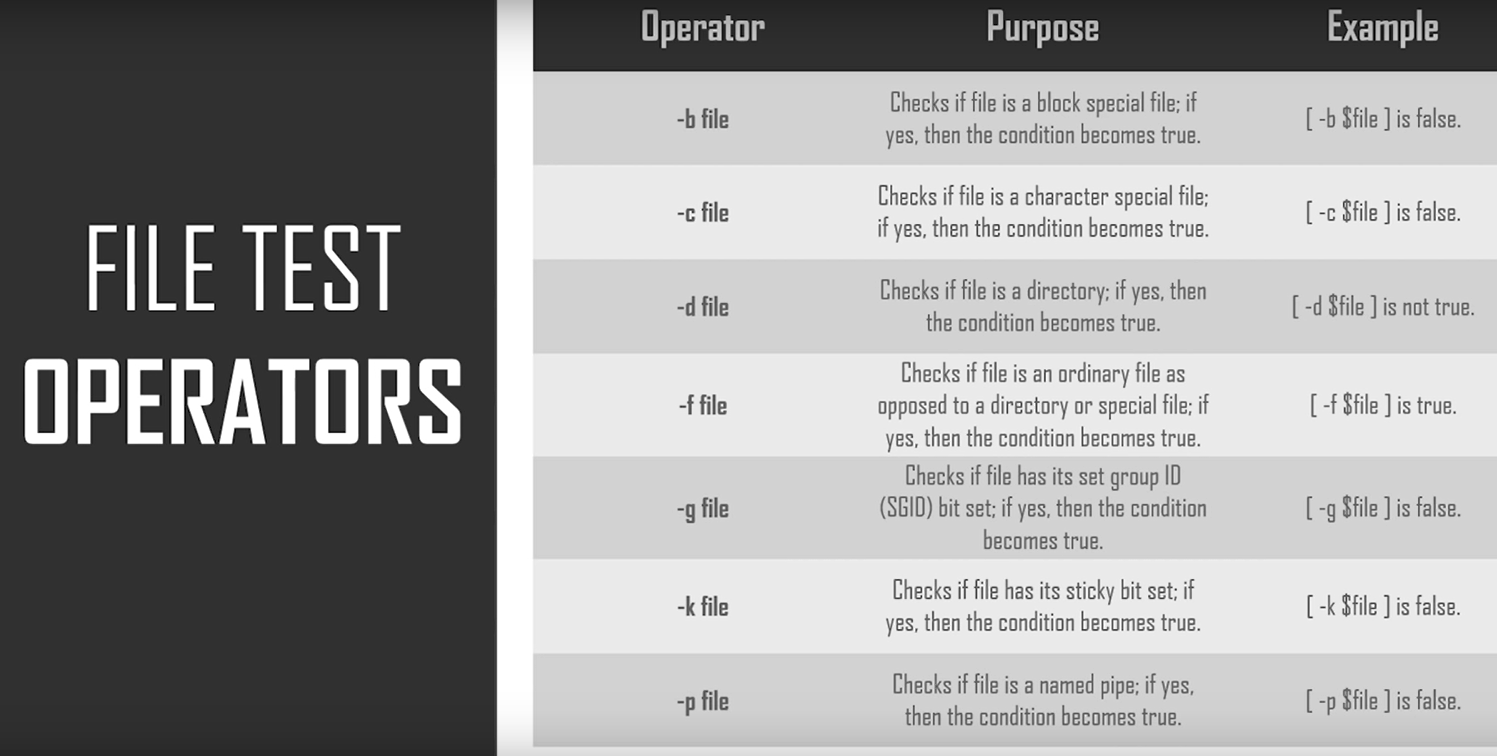
Below are screen shots of usage of each type

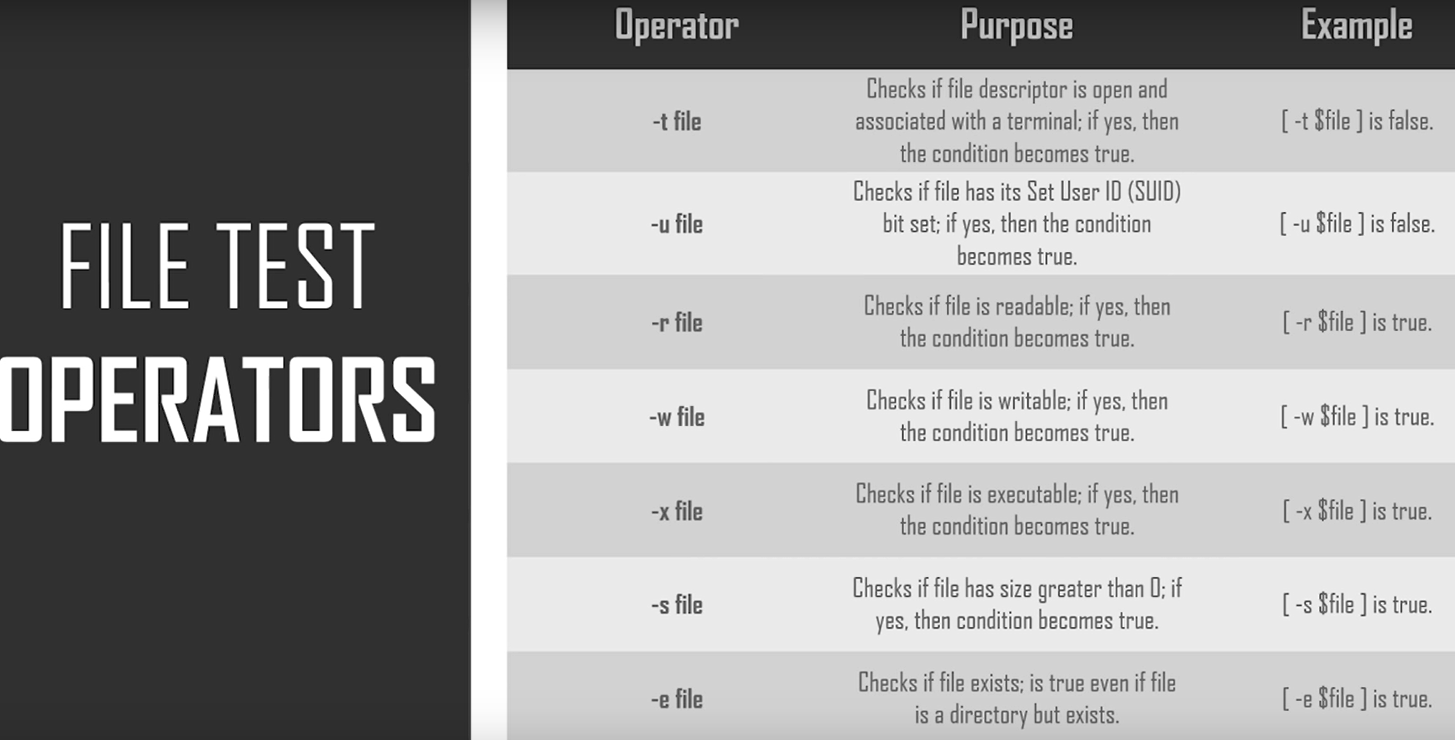












Loop in shell scripting

For loop : below is the syntax and sample script

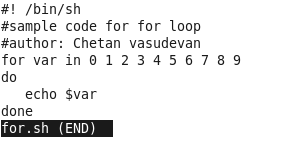
Syntax:

For var in word1,word2,word3. . . . .wordN

Do

Statement

Done



[cloudera@quickstart Shell]$ ./for.sh

0

1

2

3

4

5

6

7

8

9

While loop: while using while it executes only when your statement is true or the condition you have given /asked your code is true. Below is the syntax with an example

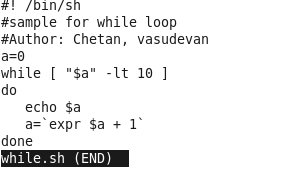
Syntax:

While command

Do

Statement

Done



[cloudera@quickstart Shell]$ ./while.sh

0

1

2

3

4

5

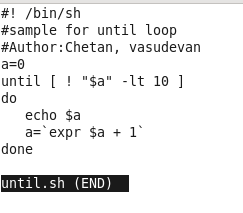
6

7

8

9

Until Loop: when you use until your code executes till your statement is true, below is the syntax and sample code



[cloudera@quickstart Shell]$ ./until.sh

0

1

2

3

4

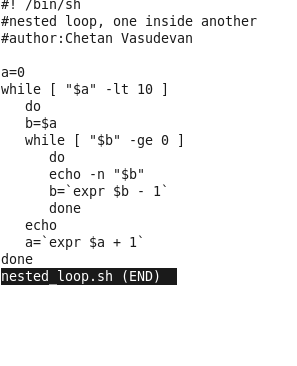
5

6

7

8

9

Nested loop : a loop inside another loop to execute multiple conditions, below is an example 

[cloudera@quickstart Shell]$ ./nested\_loop.sh

0

10

210

3210

43210

543210

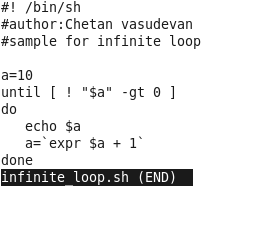
6543210

76543210

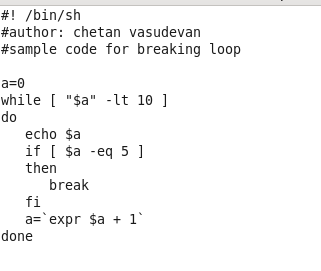
876543210

9876543210

Infinite loops: loops similar to infinite iterations which don’t stop till we stop it explicitly, below is the sample code



Break: will exit entire loop when used, below is the sample code



[cloudera@quickstart Shell]$ ./break.sh

0

1

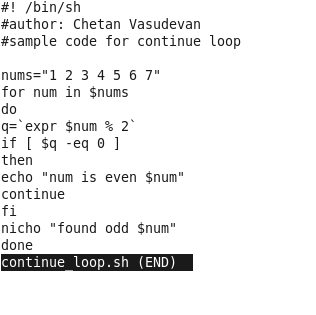
2

3

4

5

Continue: will exit only when the current iteration, below is the sample code



[cloudera@quickstart Shell]$ ./continue\_loop.sh

found odd 1

num is even 2

found odd 3

num is even 4

found odd 5

num is even 6

found odd 7

Introduction to function in shell scripting

General syntax for a basic function

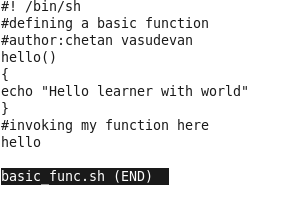
Function\_name()

{

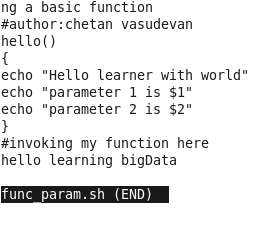
List of commands to do

}

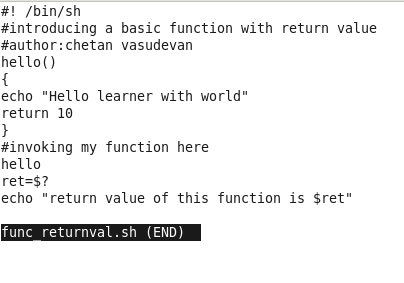
Below is a sample example



Function with parameter look like a below sample



Function with return value looks like below

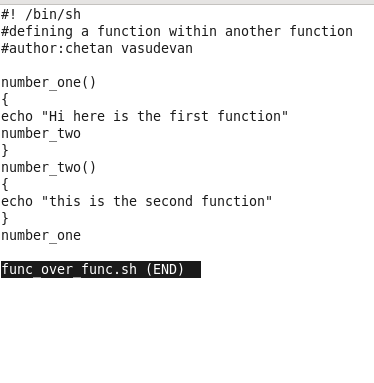


[cloudera@quickstart Shell]$ ./func\_returnval.sh

Hello learner with world

return value of this function is 10

Lets see something which is like a function calling another function, so when the primary function is called the secondary function inside the primary also gets executed. Below is the sample code



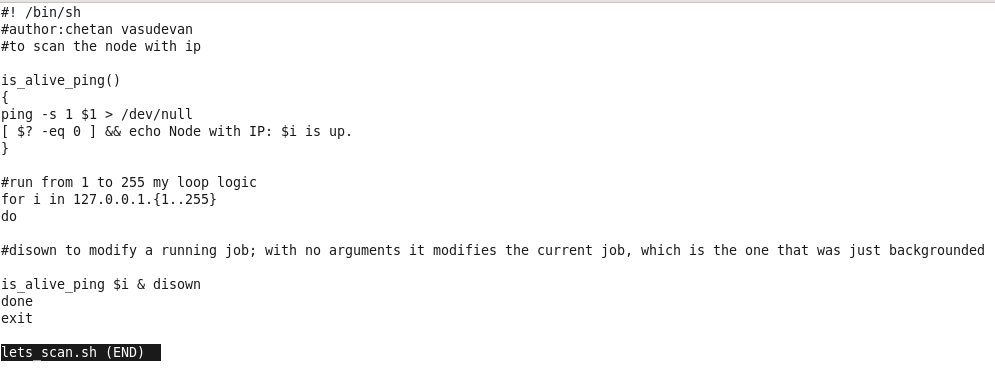
[cloudera@quickstart Shell]$ ./func\_over\_func.sh

Hi here is the first function

this is the second function

Special use cases which may be likely used or consider it as daily status cases.

Case1: to scan your node with ip and to tell if it is alive or not in other words to ping status of nodes, below is the working code sample



Case2: to check if host is up and running if it has any issues then send an email to all users/distributions.

