Introduction to Shell Scripting

Vivek Kumar Singh

June 17, 2019

Agenda

- Defining and Using Shell Scripts
- Standard Input, Output and Error
- Uing variables and Operators in Shell Scripts
- How to use conditional statements and Loops
- Using Functions to use a block of code whenever needed
- Examples and Use Cases
- Summary

Definition

Shell

Shell is a commandline interpreter. It translates commands entered by the user and converts them into a language that is understood by kernel.

Definition

Shell

Shell is a commandline interpreter. It translates commands entered by the user and converts them into a language that is understood by kernel.

Shell Script

Shell Script has list of commands in the order of execution.

Creating and Executing Scripts

create a file (vi test.sh)

```
#/bin/bash
echo "Hello Everyone"
```

Creating and Executing Scripts

create a file (vi test.sh)

```
#/bin/bash
echo "Hello Everyone"
```

make the fie executable

```
chmod +x test.sh
```

Creating and Executing Scripts

create a file (vi test.sh)

```
#/bin/bash
echo "Hello Everyone"
```

make the fie executable

```
chmod +x test.sh
```

run the Script

```
./test.sh
```

Variables

Types Of Variables

System Variables(Environment variables)
User Defined Variables
Special Variables

Variables

Types Of Variables

System Variables(Environment variables)
User Defined Variables
Special Variables

Special Variables (scriptname arg0 arg1 arg2 arg3

$0, $1, $2 scriptname, first argument, second argument
$# number of arguments
$? Exit status of Last Cmmand

Variables

Types Of Variables

System Variables(Environment variables)
User Defined Variables
Special Variables

Special Variables (scriptname arg0 arg1 arg2 arg3

$0, $1, $2 scriptname, first argument, second argument
$# number of arguments
$? Exit status of Last Cmmand

Scope of Variable

Local and Global
by default variables are global

Variable Assignment and Use

Variable Assignment

var1=45

var2=vivek

Variable Assignment and Use

Variable Assignment

var1=45
var2=vivek

Using Variables

echo "$1"
echo "my name is $var2"

Variable Assignment and Use

Variable Assignment

var1=45
var2=vivek

Using Variables

```
echo "$1"
echo "my name is $var2"
```

Taking User input

```
read name
echo "my name is $name"
```

Basic Operators

Types Of Operators

Arithmetic Operators
Relational Operators
Boolean Operators
String Operators
File Test Operators

Basic Operators

Types Of Operators

Arithmetic Operators
Relational Operators
Boolean Operators
String Operators
File Test Operators

Arithmetic Operators

```
#/bin/bash
a=10 ; b=20
echo `expr $a + $b`
echo `expr $b / $a`
```

Standard Input Input, Output and Error

Standard Input

Taking input from keyboard

Standard Input Input, Output and Error

Standard Input

Taking input from keyboard

Standard Output

Output descriptor Mainly on attached display

Standard Input Input, Output and Error

Standard Input

Taking input from keyboard

Standard Output

Output descriptor Mainly on attached display

Standard Error

Default error output device

Conditional Statements

if then else fi

If evaluates a condition, If a condition is true then if block code is executed. On false condition, else block code is executed, but its optional

Conditional Statements

if then else fi

If evaluates a condition, If a condition is true then if block code is executed. On false condition, else block code is executed, but its optional

Case esac

It allows us to execute different set of instructions against different values of a varaible

Conditional Statements

if then else fi

If evaluates a condition, If a condition is true then if block code is executed. On false condition, else block code is executed, but its optional

Case esac

It allows us to execute different set of instructions against different values of a varaible

break and continue

The break statement terminates the current loop and passes program control to the command that follows the terminated loop.

Conditional Statements

if then else fi

If evaluates a condition, If a condition is true then if block code is executed. On false condition, else block code is executed, but its optional

Case esac

It allows us to execute different set of instructions against different values of a varaible

break and continue

The break statement terminates the current loop and passes program control to the command that follows the terminated loop. continue causes a jump to the next iteration of the loop, skipping all the remaining commands in that particular loop cycle

Shell Loops

For Loop

executes commands for given set of values

Shell Loops

For Loop

executes commands for given set of values

While Loop

executes commands as long as the given condition evaluates to true

Shell Loops

For Loop

executes commands for given set of values

While Loop

executes commands as long as the given condition evaluates to true

Until Loop

execute commands as long as the given condition evaluates to false

Shell Loops

For Loop

executes commands for given set of values

While Loop

executes commands as long as the given condition evaluates to true

Until Loop

execute commands as long as the given condition evaluates to false

Infinite Loop

A loop that runs forever

Functions

Definition

Functions are blocks of code which could be reused anywhere in the code.

Functions

Definition

Functions are blocks of code which could be reused anywhere in the code.

Example: Functions and Global Variable

```
#!/bin/bsh
username=vivek
echo "Outside Function: username"
func()
{
echo" InsideFunction :username"
}
ffunc
```

Redirecting Output and Error

append and overwrite type of redirects

```
echo Hello >file1.txt # used to overwrite text to the file
echo Hello >>file2.txt # used to append text to the file
```

Redirecting Output and Error

append and overwrite type of redirects

```
echo Hello >file1.txt # used to overwrite text to the file
echo Hello >>file2.txt # used to append text to the file
```

Redirecting Standard Output

```
./scriptname.sh >logfile.txt
```

Redirecting Output and Error

append and overwrite type of redirects

```
echo Hello >file1.txt # used to overwrite text to the file
echo Hello >>file2.txt # used to append text to the file
```

Redirecting Standard Output

```
./scriptname.sh >logfile.txt
```

Redirecting Standard Error

```
./scriptname.sh 2>logfile.txt
```

Redirecting Output and Error

append and overwrite type of redirects

```
echo Hello >file1.txt # used to overwrite text to the file
echo Hello >>file2.txt # used to append text to the file
```

Redirecting Standard Output

```
./scriptname.sh >logfile.txt
```

Redirecting Standard Error

```
./scriptname.sh 2>logfile.txt
```

Redirecting Both Standard Output and Error

```
./scriptname.sh >/tmp/out.txt 2>/tmp/error.log
```

Example

Understanding Local Variables

Create a local variable and verify that its scope is local to its loop only.

Example

Understanding Local Variables

Create a local variable and verify that its scope is local to its loop only.

Solution Code

```
#/bin/bash
username=vivek
echo "Outside Function: $username"
func {
local username=vsingh
echo "Inside Function: $username" }
func
echo "Outside Function: $username"
```

Use Case : Matching User

Greeting for current user

Take username as input, If it matches with current user it should greet you, else it should say "Try Again"

Solution Code

```
#/bin/bash
echo "Please Enter username " ; read username
if [ "$username" == "$USER" ]
then
echo "Hello $username"
else
echo "Try Again"
fi
```

Use Case : Host Discovery

Host Discovery Alert

If destination server is not reachable, sen an Email

Solution Code(Run script as ./chkhost.sh host1 host2 host3)

```
#bin/bash
for i in $@
do
ping -c 1 $i >/dev/null
if [ $? -ne 0 ]
then
echo "$i is down" |mail -s "Host Down" itpc.vivek@gmail.com
fi
done
```

Summary

We Discussed

How to write small scripts
How to use variables and inputs
How to use commandline arguments in script
How to use control statements and Loops
How to use functins
How to use Redirecion

Summary

We Discussed

How to write small scripts
How to use variables and inputs
How to use commandline arguments in script
How to use control statements and Loops
How to use functins
How to use Redirecion

Scope

Further scripts could be automated using crontab

Thank You

Questions and Suggestions