Shell Scripting - Training

Mithun Technologies
devopstrainingblr@gmail.com
+91-9980923226

Agenda

- Introduction
- Why we need to learn Scripting
- Pre Requisites

Introduction

What is shell?

- Shell is a program that takes commands from the keyboard and gives them to the operating system to perform.
- Shell is a interface between user and the kernel.
- On most Linux systems a program called bash (which stands for Bourne Again SHell, an enhanced version of the original Unix shell program, sh, written by Steve Bourne) acts as the shell program.
- There are other shell programs as follows.

  Korn Shell (ksh)

  Bourne Again Shell (bash)

  C Shell (csh)

  Tiny Shell(tcsh)

  Z Shell (zsh)

Introduction cont..

The /etc/shells is a text file which contains the full pathnames of valid login shells.
Type the following command to see list how many shells are installed on your Linux or Unix server.

cat /etc/shells

```
[mithun@mithuntechnologies ~]$ cat /etc/shells
/bin/sh
/bin/bash
/sbin/nologin
/usr/bin/sh
/usr/bin/bash
/usr/sbin/nologin
[mithun@mithuntechnologies ~]$
```

```
/bin/sh
/bin/bash
/sbin/nologin
/bin/tcsh
/bin/csh
/bin/ksh
/bin/dash
```

Introduction cont..

How to change and find the shell type?
/bin/csh
/bin/ksh
/bin/sh
/bin/bash

To check what which is the current shell, use one of the following command

```
echo $0
ps -p $$
echo $SHELL
```

Introduction

What is Shell Scripting?

In the simplest terms, a shell script is a file containing a series of commands. The shell reads this file and carries out the commands as though they have been entered directly on the command line.

Normally the shell scripts has the file extension .sh

Why we need to learn Scripting

- Automate your regular jobs.
- Taking database backups
- Monitoring several server resources like cpu utilization and memory utilization …
- Portable (It can be executed in any Unix-like operating systems without any modifications)

Pre Requisites

- Linux/Unix/AIX server.
- Knowledge on Linux commands with various options.
- Basic knowledge of programming.

Steps to write shell script

Step 1) Use any editor like vi or gedit to write shell script.
Syntax: vi << Script file name >>

Example:
#vi hello.sh // Start vi editor

Step 2) After writing shell script set execute permission for your script as follows
Syntax: chmod << permission >> <<script file name >>

Examples:
chmod +x script file name

Step 3) Execute your script as
Syntax: bash script file name
sh script file name
./script file name
. script file name

First Shell script

File Name: hello.sh

```
#! /bin/bash
echo “Hello Welcome to Shell script!”
```

The first line is called a shebang or a "bang" line. It is nothing but the absolute path to the Bash interpreter.

File Naming conventions

1) A file name can be a maximum of 255 characters
2) The name may contain alphabets, digits, dots and underscores
3) System commands or Linux reserve words can not be used for file names.
4) File system is case sensitive.
5) Some of valid filenames in Linux are

bhaskar.sh
mithun.sh
Bhaskar.sh
Mithun.sh
Mithun_08112013.sh
Bhaskar05.sh

Comments

- Comments are used to escape from the code.
- This part of the code will be ignored by the program interpreter.
- Adding comments make things easy for the programmer, while editing the code in future.
- Single line comments can be do using #

```
# This is single line comment.
```

- Multilane comments can be do using HERE DOCUMENT feature as follows

```
<<COMMENT1
   your comment 1
   comment 2
   blah
COMMENT1
```

Variables

A **variable** is a character string to which we assign a value. The value assigned could be a number, text, filename, device, or any other type of data.

There are two types of variables in Linux shell script.
Those are 1) System variables
2) User defined variables.

System Variables

- Created and maintained by Linux bash shell itself. This type of variable is defined in CAPITAL LETTERS.
- There are many shell inbuilt variables which are used for administration and writing shell scripts.
- To see all system variables, type the following command at a console / terminal:
  env or printenv

User defined Variables

Created and maintained by user. This type of variable defined may use any valid variable name, but it is good practice to avoid all uppercase names as many are used by the shell.

Ex: name=”Mithun Reddy“
id=08112012

Note: No space between variable name and value.

Commandline Arguments

During shell script execution, values passing through command prompt is called as command line arguments.

For example while running a shell script, we can specify the command line arguments as “sh scriptfile.sh arg1 arg2 arg3”

While using command line arguments follow the below important points.

- We can specify n number of arguments, there is no limitation.
- Each argument is separated by space.

Escape character

When using echo -e, you can use the following special characters:

\\ backslash
\a alert (BEL)
\b backspace
\c suppress trailing newline
\f form feed
\n new line
\t horizontal tab
\v vertical tab

String

String is a zero or more characters enclosed in single or double quotes.
We use single or double quotations to define a string.

String contd…

Find the length of the string

FileName: string_length.sh

```
#! /bin/bash

string_var="Hi Team, My name is Mithun Reddy L, working in IBM, Manyatha Techpark "

echo The string value is:  ${string_var}

echo The length of the string is:  ${#string_var}
```

String contd…

Find the substring from the string

FileName: string_substring.sh

```
#! /bin/bash

string_var="Hi Team, My name is Mithun Reddy L, working in IBM, Manyatha Techpark"
echo ${string_var:8}
echo ${string_var:20:14}
```

Arithmetic Operations

We use the keyword " expr " to perform arithmetic operations.

Syntax:
expr op1 math-operator op2

FileName: arithmetic_operations.sh

```
expr 3 + 2
expr 3 - 2
expr 10 / 2
expr 3\* 2
expr 20 % 3
echo addition of 3 and 2 is : `expr 3 + 2`
```

Note: There must be spaces between the operators and the expressions. For example, 3+2 is not correct; it should be written as 3 + 2.
Complete expression should be enclosed between ` `, called the inverted commas.

User Interaction using read command

read command is used to get the input from the user (Making scripts interactive).

File Name: readName.sh

```
#!/bin/bash
#Author: Bhaskar Reddy Lacchannagari.
#Date: 7th June 2013.

echo "Please enter your name:"
read userName

echo The name you entered is $userName
```

read command contd..

File Name: readMultipleValues.sh

```
#!/bin/bash
#Author: Bhaskar Reddy Lacchannagari.
#Date: 7th June 2013.

echo "Please enter DevOps Tools/Techniques:"
read devOpsTools

echo The DevOps Tools/Techniques you entered are $devOpsTools
```

read command contd..

File Name: readWithOptions2.sh

```
#Author: Bhaskar Reddy Lacchannagari.
#Date: 7th June 2013.

echo 'Enter DevOps Tools: '
read -a technologies
echo "DevOps Tools are:
${technologies[0]},${technologies[1]},${technologies[2]},${technologies[3]},${technologies[4]}"
```

read command contd..

File Name: readWithOptions3.sh

```
#Author: Bhaskar Reddy Lacchannagari.
#Date: 7th June 2013.

echo 'Enter DevOps Tools: '
read
echo "DevOps Tools are: $REPLY"
```

read command contd..

File Name: readWithOptions1.sh

```
#Author: Bhaskar Reddy Lacchannagari.
#Date: 7th June 2013.

read  -p 'Enter User Name ' userName
read -sp 'Enter the password ' password

echo '      '
echo 'You entered username is: '$userName
echo 'You entered password is: '$password
```

Input and Output Redirection

Using shell scripts, we can redirect - the output of a command to a file
or
- redirect an output file as an input to other commands.
In Shell script there are mainly 3 types of redirect symbols as follows.
1. > Redirect standard output

Example:
ls > ls-file.txt

The above command will redirect the output of the " ls " to the file " ls-file.txt ".
If the file " ls-file.txt " already exist, it will be overwritten. Here you will loose the existing data.

Input and Output Redirection contd..

2. >> Append standard output

Example:
date >> ls-file.txt

The output of the date command will be appended to the file " ls-file.txt ".
In this case you will not loose any data. The new data gets added to the end of the file.

3. < Redirect standard input

Example:
cat < ls-file.txt

This redirection symbol takes input from a file.
In the above example the cat command takes the input from the file " ls-file.txt " and displays the "ls-file.txt" content.

sh comm.sh 1> stdoutput.log 2>stderror.log
sh comm.sh > log.txt 2>&1
sh comm.sh > log.txt 1>&2

Control commands – if

Syntax:

if condition
then
    Display commands list if condition is true.
else
    Display commands list if condition is false.
fi

Note: if and **then** must be separated, either with a << new line >> or a semicolon (;).
Termination of the if statement is **fi**.

Control commands – for

Syntax:

```
for (condition )
do
        execute here all command/script  until the condition is
        not satisfied.(And repeat all statement between do and done)

done
```

Control commands – while loop

Syntax:

```
initialisation
while [ condition ]
        do
            command1
            command2
            command3
            ..
            ....
        done
```

Control Commands - switch case

The switch case statement is good alternative to multilevel if-then-else-fi statement. It enables you to match several values against one variable. It's easier to read and write.

Syntax:

```
case $variable-name  in
    pattern1)   command
                ...
                ..
                 command;;
    pattern2)   command
                ...
                ..
                  command;;
    patternN)   command
                ...
                ..
                command;;
    *)          command
                ...
                ..
                command;;
esac
```

Functions

When your scripts start to become very large, you may tend to notice that you are repeating code more often in your scripts. You have the ability to create functions inside of your script to help with code reuse. Writing the same code in multiple sections of your script can lead to severe maintenance problems. When you fix a bug in a section of code you need to be sure that all sections of code that are repeated will also have those fixes. A function is a block of code that can be called from other parts of your script. It can have parameters passed to it as if it were a separate script itself. As an example, we will create a function called log it, which will take two parameters, a level and a message. The level will be a number between 1 and 3 that will indicate the severity of the log message. The level of messages that you want to view will be passed in the command line of the script.

Syntax:

function_name() {

Commands to be execute here...

}

Pipes

A pipe is nothing but a temporary storage place where the output of one command is stored and then passed as the input for second command. Pipes are used to run more than two commands (Multiple commands) from same command line.Pipelines connect the standard output of one command directly to the standard input of another. The pipe symbol (|) is used between the commands:

Example: Creating a new file " friends ".
vim friends.txt
Mithun
Ruthik
Mourya
Shishir

Now issue the following command:
#sort friends.txt | tr "[a-z]" "[A-Z]" > FRIENDS.txt
#cat FRIENDS.txt

Questions ?

Thank you

Mithun Technologies
+91-9980923226
devopstrainingblr@gmail.com