Persistent

Bash Shell Scripting

Shell Scripting Basics – Part 1

Persistent University

Training Goals

To study basics of Bash shell scripting and use it to automate day to day activities

- The Training covers following topics:
- Module 1: Shell Scripting Basics
- Module 2: Shell Arithmetic
- Module 3: Shell Control Structures
- Module 4: Arrays
- Module 5: Functions
- Module 6: Advanced Shell Scripting
- Module 7: SED
- Module 8: Grep and Regular expressions
- Module 9: AWK

Contents

Shell Scripting Basics – Part I

At the end of this module , you will be able to understand:

Shell Basics

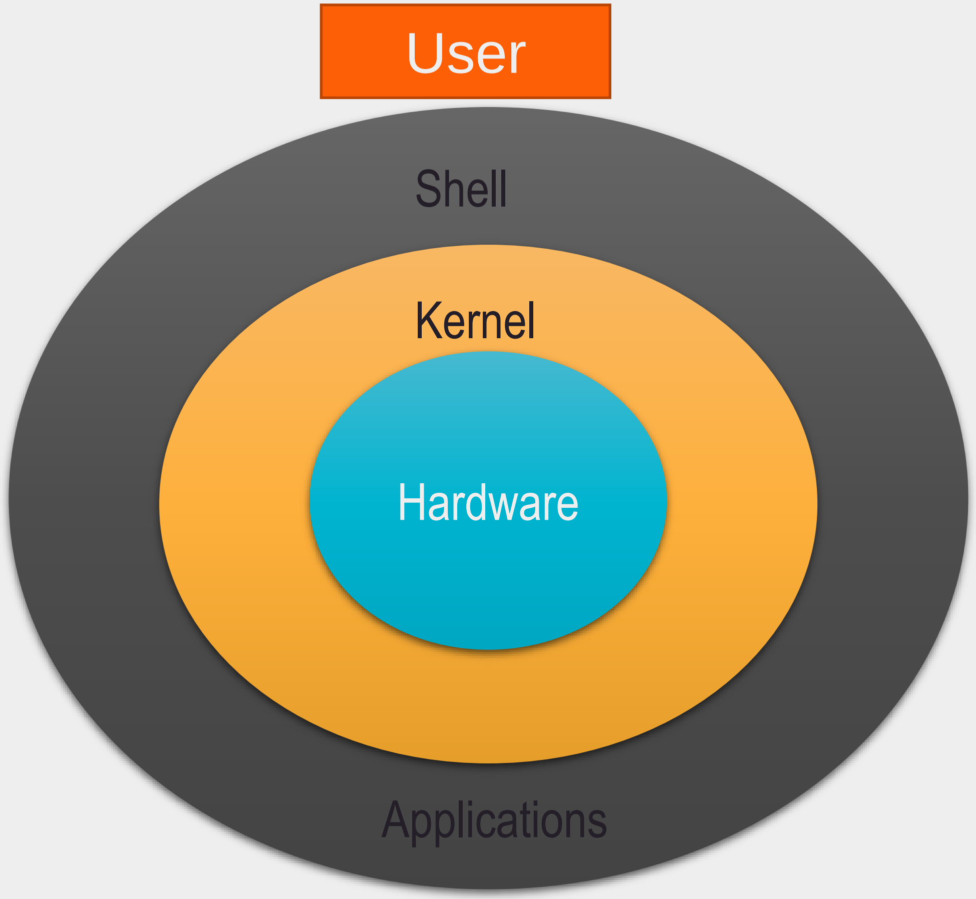
What are Shell Variables?

What are positional parameters and how to access those?

How to write sample Shell Script?

Shell

- What is Shell and how it interacts with kernel?



Shell Basics

•

- A shell is the command prompt within Linux where you can type commands
- A shell is a program that acts as the interface between the user and the system, allowing the user to enter commands for the OS to execute
- It resembles a Windows command prompt, but is much more powerful
- On Unix, it’s quite feasible to have multiple shells installed, with different users able to pick the one they prefer
- Most of the shells are derived from the original Bourne shell

Shell types

- sh (Bourne) - the original shell from early version of Unix
- csh, tcsh, zsh - the C shell, and its derivatives
- ksh, pdksh - the Korn shell and its public domain version
- bash - Bourne Again Shell, is open source and is ported to most of the Unix variants. It is similar to Korn shell
- To find available shells : $cat /etc/shells

Advantages of bash shell

- Bash shell is extremely popular shell. Its advantages are -
  - It offers environment variables to configure every aspect of your user experience within the shell
  - It supports command history & command completion
  - It offers built-in arithmetic functions
  - Wildcard expressions help in finding the exact name of a program or see all the programs with a specific string in the file name.
  - Command line editing helps you use Emacs and vi commands to make edits at the prompt
  - Bash has a long list of built-in commands that make machine administration simple and much easier
  - Scripts for bash are compatible with older Bourne shell

What is shell scripting

- A shell script is a plain-text file that contains shell commands.
- To be executable, a shell script file must meet some conditions -
  - The file must have a special first line that names an appropriate command processor. In this learning material we will be using bash processor:
    - #!/bin/bash
  - If this example doesn't work, you will need to find out where your Bash shell executable is located and substitute that location in the above example. Here is one way to find out:
    - $ whereis bash
  - The file must be made executable by changing its permission bits. An example:
    - $ chmod +x (shell script filename)
- A shell script file may optionally have an identifying suffix, like ".sh"

Bash Shell Scripting – Contd ..

- Shell script starts with shebang “#!”
- Scripts are executed like programs, and they need to have the proper execute permissions
  - chmod +x ~/somefolder/script_name
- Comment line start with a #
- Command substitutions are done with single tick marks ”`” or parentheses “( )”
- Use any editor like vi to write shell script

Always be sure your script name doesn’t conflict with Shell commands

How to execute a shell script?

- How to set execute permissions to shell script?
  - Examples:

    $ chmod +x script-name

    OR

    $ chmod 755 script-name
- Bash Shell Script - Execution ways
  - bash script-name
  - sh script-name
  - ./script-name
  - . ./script-name
  - source script-name

Writing first shell script

- How to write first shell script -
  - Choose a text editor you want to use
  - Run your choice of editor and type the following lines:
  - #!/bin/bash
  - echo "Knowledge is Power"
  - Save the file in the current working directory as "myscript.sh"
  - Move from the text editor to a command shell
  - From the command shell, execute below command to assign execute permissions to script :
  - $ chmod +x myscript.sh
- To execute the script, run following command :
  - $ ./myscript.sh
  - Knowledge is Power

Why to write shell script

- Advantages of writing shell scripts :
  - Shell script can take input from user, file and output them on screen
  - Useful to create our own commands
  - Save lots of time
  - To automate some task of day today life
  - System Administration part can be also automated

Variables

- Variable is character string to which we assign a value
- A variable name :
  - Must begin with Alphanumeric character or Underscore character
  - Contains a..z, A..Z, 0..9 or _
  - Is case sensitive
- User Defined Variables (UDV)
  - Syntax :- variable-name=value
  - myVar=10
- No spaces on either side of the = sign

Variables contd..

NAME=“Sara“

readonly NAME

NAME=“Anna”

- How to access variable
  - myVar=“HELLO WORLD!!”
  - echo $myVar
- Exporting variable
  - export myVar=“HELLO WORLD!!”
- How to set variable as “Readonly” variable?

Variables contd..

- System Variables
  - Also known as environment variables
- To display, use **env** or **printenv** commands
  - HOME, PATH, USER, HOSTNAME, PWD etc.
- To print, use $
  - echo $HOME

Do not modify value of system variable. This can sometimes creates problem

Variable - Examples

- Valid variable names
  - myVar
  - My_Var
  - VAR_1,
  - var_1
- Invalid variable names
  - _2_VAR
  - -VARIABLE
  - Var-1
  - VAR_A!

Variable - Examples

- You can define NULL variable as as follows (NULL variable is variable which has no value at the time of definition) For example.
  - $ vech =
  - $ vech =“”
- Try to print it's value by issuing following command
  - $ echo $ vech.
- Nothing will be shown because variable has no value that is NULL variable.
- Do not use question mark ,asterisk etc, to name your variable names

Variables contd..

- Examples of system variables

$BASH_VERSION	: version of bash installed on system
$BASH	: path to bash interpreter
$HOME	: path of home directory
$PWD	: current working directory
$USERNAME	: currently logged in user
$OSTYPE	: operating system type
$UID	: current user’s identification number
$GROUPS	: group current user belongs to

Variables contd..

- Example:
  - #!/bin/bash
  - echo “Hello, $USER. I wish to list some files of yours"
  - echo “Files in the current directory, $PWD"
  - ls   # list files

Reading User Input

- read command
  - Example:
  - echo "Please enter your name:“
  - read name
  - echo "Welcome $name !!!“
  - echo -n "Enter some text > "
  - read text
  - echo "You entered: $text"

echo command

- echo command:
  - Echo command is used to display text or value of a variable
  - Syntax of the echo command is -

    ```
    echo [options] [string, variables...]
    ```

  - Options of the echo command are –

    -n   Do not output the trailing new line.

    -e   Enable interpretation of the following backslash escaped characters in the strings:

    \a alert (bell)

    \b backspace

    \c suppress trailing new line

    \n new line

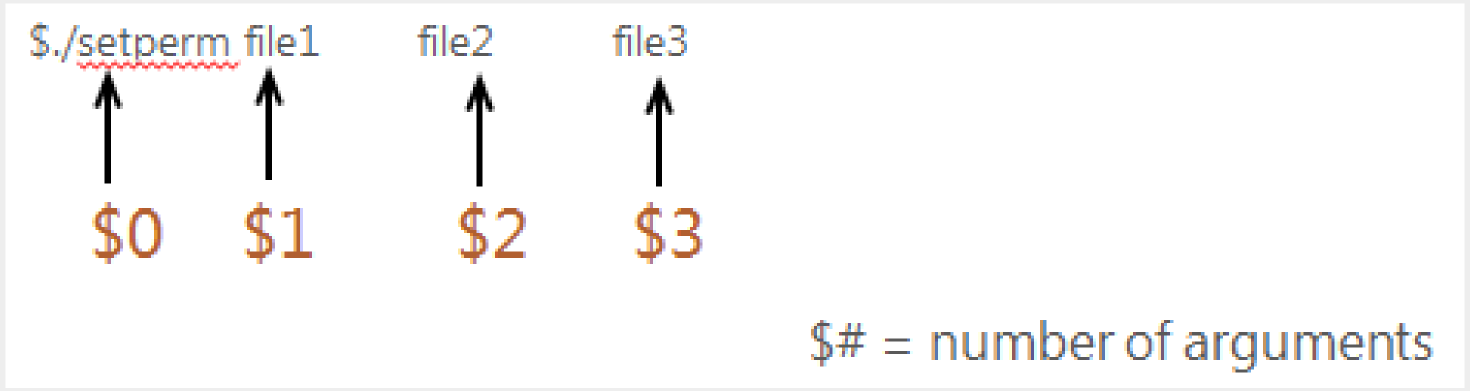
    \r carriage return

    \t horizontal tab

    \\ backslash

Positional Parameters

- Positional parameters are variables defined by the shell
- Positional parameters are created when script is invoked with parameters
  - $0 is the name of the script itself
  - $1, $2 …. The parameters given to the script
  - After $9, the arguments must be enclosed in brackets to access its values
  - E.g ${10}, ${11}, ${12}



Positional Parameters

- **Positional parameters are command line arguments passed to a script**
  - $#   : Total number of arguments
  - $0   : Command or the script name
  - $1,$2,…$N   : First, second and N th args respectively
  - $* : Single string that consist of command line arguments starting from $1
  - $@   : Array of “$1” “$2” .. “$N”
- **Example : ./myScript.sh arg1 arg2**
  - $# : 2
  - $0 : ./myScript.sh
  - $1 : arg1 , $2 : arg2
  - $* : arg1 arg2

Positional Parameters contd..

- shift operates on positional parameters
- Useful when number of arguments not known in advance
- shift n
  - **Example:**
  - while [“$1” != “”]; do
  - echo “Now you have $# parameters”
    - shift
    - done

Quiz

- In shell script, command substitutions are done with ____________
- If you try to assign value to readonly variable, it gives error ________
- To list out system variables, use commands _______
- To print total number of command line parameters ____ is used
- _____ Variable prints present working directory of user.

Quiz Answers

- In shell script, command substitutions are done with **Single tick or parenthesis ().**
- If you try to assign value to readonly variable, it gives error **“This variable is read only”.**
- To list out system variables, use commands **env and printenv.**
- To print total number of command line parameters **$#** is used
- **PWD** Variable prints present working directory of user.

Assignments

- Q1: Write a script that will accept 3 arguments and using those create a single string separated by colon.
  - Input : ./test.sh abc pqr xyz
  - Output: abc:pqr:xyz

- Q2: Write a script to accept as firstname, middlename and lastname from user. Display full name in the format - lastname firstname middlename.

Assignment Solutions

- **Assignment 1:**
  - str="$1:$2:$3"
  - echo "Complete string is:- $str“

- **Assignment 2:**
  - read -p "Enter firstname: " firstname
  - read -p "Enter middlename: " middlename
  - read -p "Enter lastname: " lastname
  - fullname="$lastname $firstname $middlename"
  - echo "Fullname : “ $fullname

Summary

- In this module, we have learnt about Shell scripting basics.
- Now, you should be able to answer following questions:
  - What is Shell?
  - How to write shell script?
  - What is Shell variable and how to use them?
  - What are positional parameters and how to access those?

Reference material

- http://www.freeos.com/guides/lsst/
- http://www.howtogeek.com/67469/the-beginners-guide-to-shell-scripting-the-basics/
- http://www.tutorialspoint.com/unix/unix-what-is-shell.htm

Key contacts

- **Persistent Interactive**

  Bhimashankar Gavkare

  bhimashankar_gavkare@persistent.com

Persistent

Thank You !!!

Persistent University