Linux Shell Programming

Session 4

Agenda

- Command line arguments
- Arithmetic in shell scripts
- Read and echo commands in shell script (I/O)
- Taking decision

  - if, elif, else

  - test and []

  - switch case

Oh Shell is programming language as well

- How?

  - Interpretive ability

  - In built commands for making decisions

- First Shell script - don’t worry about extension but good to have one
- First line in shell script
- How to run it ?

  - Make it executable

  - Use shell to invoke shell script

Shell script

- vi hello.sh

```
#!/bin/bash
#Above line is called sheline
echo "Let's say hello from shell script"
echo "Some variables in shell"
course="DAC Aug 2015"
module="Linux and OS"
echo "Wlecome to $module module of $course course"
echo "Exiting ..."
exit 0
```

- Execution

```
chmod +x hello.sh
./hello.sh
```

OR

```
bash hello.sh
```

Send something to the script - command line arg

- Variables in shell script

$1..$9	Command line arguments or positional parameter
$#	Number of arguments specified in command line
$0	Name of the executed command/shell
$*, “$*”	Complete list of positional parameters or command line arguments as a single string
$@, “$@”	Each quoted string treated as separate argument (recommended for use over $*)
$?	Exit status of last command
$$	PID of current shell

Hello.sh with command line arguments

- Hello-arg.sh

```
#!/bin/bash

echo "I am ivoked with $0 name"
echo "Welcome to $1 module of $2 course"
echo "All arguments as single string $*"
echo "All arguments as separate strings $@"
exit 0
```

- Execution

```
./Hello-arg.sh "Linux and OS" "DAC Aug 2015"
```

Arithmetic Evaluation in Shell

- addition=1+2

  echo “$addition”

- Ways of performing arithmetic (Shell does not have any built in support)
  a. expr
  b. let
  c. $(())

```
#!/bin/bash

a=10;b=20;c=30;d=40;

addition=`expr $a + $b`     # $ is compulsory for variables, should have space around operator(+), no space around =

let sum=a+b                 # No space around operator or =, $ is not mandatory before variable name

total=$((a+b))              # $ before var name is not mandatory, no space around = and +
```

Execution status of command

- <command invocation>

  - echo $? ## $? contains exit status of last executed command

- Example

  grep “Sachin Tendulkar” dac-aug-2015-list.txt

  echo “$?”

  - 0 if pattern found else non zero

- Wrong Usage

  grep “Sachin Tendulkar” dac-aug-2015-list.txt

  echo “Performed pattern search using grep now checking exit status”

  echo “$?”

Conditional operators

- &&

  <command1> && <command2>

  - If execution of command1 is successful (0 exit status) then command2 is executed else command2 will not be executed

- ||

  <command1> || <command2>

  - If execution of command1 is unsuccessful(non 0 exit status then command2 is executed else command2 will not be executed

Example of && and ||

- conditional.sh

```
#!/bin/bash

grep “Sachin Tendulkar” dac-aug-2015-student-list.txt || echo “Pattern not found or file not present” && exit 2

echo “Pattern found”

exit 0
```

- Execution

```
chmod +x conditional.sh

./conditional.sh
```

if conditional

if, else	**only if**	**if, elif, else (nested)**
if command is successful then execute commands else execute commands fi	if command is successful then execute commands fi	if command is successful then execute commands elif command is successful then execute commands else execute commands fi

if conditional

- Example - if-else.sh

```
#!/bin/bash
 if cat file.txt; then
      echo “Contents of file file.txt are displayed successfully”
 else
      echo “Unable to display contents of file successfully”
 fi
```

Testing expressions

- The if statement can only check the exit status of command

- It can not evaluate truthness of expressions on its own so we need to take help from some other commands (test and [])

- Requirement
  a. Comparing two variables (integer, tricky as all variables in shell are string)
  b. Compare strings (checking emptiness of string)
  c. Test existence and type of file

test and []

Sr No	Expression type	Using test	Using []	Example using if

1	**Numerical**
	-eq => Equal to
	-ne => not equal to
	-gt => Greater than
	-ge => Greater than equal to
	-lt => Less than
	-le => Less than or equal to
		test $var1 -eq $var2
		test $var1 -ne $var2
		test $var1 -gt $var2
		test $var1 -ge $var2
		test $var1 -lt $var2
		test $var1 -le $var2
			[$var1 -eq $var2]
			[$var1 -ne $var2]
			[$var1 -gt $var2]
			[$var1 -ge $var2]
			[$var1 -lt $var2]
			[$var1 -le $var2]
				#!/bin/bash
				var1=10;var2=20;
				if test $var1 -eq $var2 ; then
				  echo “var1 and var2 are equal”
				elif [$var1 -gt $var2]
				then
				  echo “var1 is greater than var2”
				else
				  echo “var1 is less than var2”
				fi

test and []

Sr No	Expression type	Using test	Using []	Example using if

2	**String**
	= => equal to
	!= => not equal to
	-n => not null string (unary)
	-z => null string (unary)
		test $str1 = $str2
		test $str1 != $str2
		test -n $str1
		test -z $str2
			[$str1 = $str2]
			[$str1 != $str2]
			[-n $str1]
			[-z $str1]
				#!/bin/bash
				str1=”Hi”;str=”hello”
				if test $str1 = $str2
				then
				  echo “str1 and str2 are equal”
				else
				  echo “str1 and str2 are not equal”
				fi
				if [-n $str1];then # [! -z $str1];then
				  echo “str1 is not null”
				else
				  echo “str1 is null”
				fi

test and []

Sr No	Expression type	Using test	Using []	Example using if

3	**File**
	-f => file exists and is regular
	-r => file exists and is readable
	-w => file exists and is writable
	-x => file exists and is executable
	-d => file exists and is directory
	-s => file exists and has a size greater than zero
	(Above tests are unary)
	(Below test are binary)
	-nt => left hand file is newer than right hand file
	-ot => left hand file is older than right hand file
	
	and many more
		
		test -f $file
		test -r $file
		test -w $file
		test -x $file
		test -d $file
		test -s $file
		
		test $f1 -nt $f2
		
		test $f1 -ot $f2
			
			[-f $file]
			[-r $file]
			[-w $file]
			[-x $file]
			[-d $file]
			[-s $file]
			
			[$f1 -nt $f2]
			
			[$f1 -ot $f2]
				#!/bin/bash
				file=”regular.txt”
				dir=”/home/shivraj”
				if [-e “$file”];then
				  echo “File exists”
				fi
				if test -f “$file” ;then
				  echo “File is regular”
				fi
				if [-d “$file”];then
				  echo “File is directory”
				fi
				if [-r “$file” -a -w “$file”]
				then
				  echo “File is both readable and writable”
				fi

The case conditional (don’t say switch in shell)

- Multiway branching (preferable over nested if statements and for string tests)
- Syntax

```
case expression in
    pattern1) commands1 ;;          ## check for double semi colon
    pattern2) commands2 ;;
    pattern3) commands3 ;;
    ...
    *) commands ;;                  ## if no pattern matched
esac                                ## Reverse of case
```

Example of case

```
#!/bin/bash
# menu.sh :    Uses case to offer 5-item menu
echo -e “           Menu\n
 1.    List all of files\n2. Processes of user\n3. Today’s Date\n4. Users of
       System\n5. Quit to shell”
echo -e “Enter your option : \c”
read choice
case “$choice” in
 1)    ls -l ;;
 2)    ps -f ;;
 3)    date ;;
 4)    who ;;
 5)    exit ;;
   *) echo “Invalid option”        # ;; are not really required for las option
esac
```