



## Linux Shell Script



### Shell Script : Introduction

- Shell script is collection of commands along with programming constructs.
- Shell script syntax will differ from shell to shell.
- Shell scripts are interpreted by shell (interpreter - line by line).
- Speed is slower.
  
- Comments in shell script begin with # symbol.
  
- **echo command**
  - -e : enable escape sequences e.g. \n, \t, ...
  - -n : no newline after echo.
  
- **shebang line (#!/bin/bash)**
  - Line 1 of shell script should contain name of shell program which will execute that script followed by #!.
  - While running script on terminal (./demo01.sh), OS reads first line and load corresponding shell program, which in turn execute that shell script.



## Shell Script : Variables

### • Shell variables

- Shell scripts are type-less. There is no concept of data types.
- Also no need to declare variables before using them.

### • Assign value to variable

- varname=value

### • Read value from variable

- \$varname

### • Assign output of command to variable

- varname='command ...'
- varname=\$(command ...)

### • To perform arithmetic – expr and bc

### • Environment variables

- Few variables are initialized from values in the environment.
- Normally these variables are in uppercase to distinguish from user defined variables.
- Variables created depends on your personal configuration.

#### • e.g.

- \$HOME – gives home directory
- \$PATH – path of all executables
- \$USER – gives user name
- \$SHELL – gives which shell is currently running.



## Shell Script : Conditions

- Test conditions and perform different actions based on those decisions.
- For checking conditions you can use following two syntaxes:
  - test condition
  - [ condition ]
- Condition types that can be used are of three types:
  - String comparison
  - Arithmetic comparison
  - File conditions

String Comparison	Result
str1 = str2	True if str1 and str2 are equal
str1 != str2	True if str1 and str2 are not equal
-n str	True if the str is not null
-z string	True if the str is null (empty)

Arith Comparison	Result
exp1 -eq exp2	True if equal
exp1 -ne exp2	True if not equal
exp1 -gt exp2	True if exp1 is greater than exp2
exp1 -ge exp2	True if exp1 is greater or equal exp2
exp1 -lt exp2	True if exp1 is less than exp2
exp1 -le exp2	True if exp1 is less or equal exp2

File Conditionals	Result
-e file	True if file exists
-f file	True if file is regular file
-d file	True if file is directory
-r file	True if file readable
-w file	True if file is writable
-x file	True if file executable



## Shell Script : Control Structures

- if

```
if [ condition ]
then
    # ...
fi
```

```
if [ condition ]
then
    # ...
elif [ condition ]
then
    # ...
else
    # ...
fi
```

```
if [ condition ]
then
    # ...
else
    # ...
fi
```

```
if [ condition ]
then
    # ...
else
    if [ condition ]
    then
        # ...
    else
        # ...
    fi
fi
```

- case

```
case $var in
c1|const1|case1)
    # ...
;;
c2|case2)
    # ...
;;
c3)
    # ...
;;
*)
esac
```

- for

```
for var in collection
do
    # ...
done
```

- while

```
# initialization
while [ condition ]
do
    # ...
    # modification
done
```

- until

```
# initialization
until [ condition ]
do
    # ...
    # modification
done
```



## Shell Script : Functions

- Function without return

```
# function definition:
function fn_name()
{
    # args are accessed as $1, $2, $3, ...
    #
}

# function call:
fn_name arg1 arg2 arg3
```

- Function with return

```
# function definition:
function fn_name()
{
    # args are accessed as $1, $2, $3, ...
    #
    echo result
}

# function call:
var=$(fn_name arg1 arg2 arg3)
```



## Shell Script : Positional Parameters

- Positional parameters (like command line arguments in C)
- While executing shell script on command line, we can pass additional information called as "**positional parameters**".
- terminal> **./dem.sh one two three four**
- To access positional parameters in the script: **\$1 \$2 \$3 \$4 ... \$9**
- List of all positional parameters: **\$\***
- Shell script name: **\$0**
- Number of positional parameters: **\$#**
- **shift N** command is used **to skip N parameters from left**
- This will enable access to the next parameters.
- **N+1** parameter will become **\$1**
- **N+2** parameter will become **\$2**



## Shell Script : Array

- Array is collection of values
  - arr=(1,2,3,4,5)
- To print all values:  **\${arr[\*]}**  or  **\${arr[@]}**
- To print individual element  **\${arr[i]}**
- To print number of elements:  **\${#arr[\*]}**
- declaration is optional
  - declare -a arr



## Shell Script : .bashrc and .profile

- .bashrc file
  - .bashrc is shell script that is executed when a new CLI bash shell is started.
  - We can add commands to be executed when new shell starts.
  - Example:
    - alias c=clear
    - echo "Welcome to bash!"
    - export PATH=/some/path:\$PATH
  - To edit the file
    - terminal> vim ~/.bashrc
    - Add your commands to the end of file.
  - These changes will be visible when new shell is started.
  - Close current terminal and open new terminal.
- .profile file
  - .profile is shell script that is executed when new login shell is started.
  - This will run for tty terminals or gui terminals.
  - When we need to execute some commands when any new login is done, those commands should be written in .profile.



Thank you!

