GLOBAL EDGE
Intelligence Of Things™

Basic Shell Programming

Shubham Talekar
Jyothi A.

Contents

1. Introduction

- What is shell scripting ?
- Why we require shell scripting ?
- To run :

make executable: `$ chmod +x <script_filename>`
`$ ./<script_filename>`

To run in command line:
`$ sh <script_filename>`

2. User Input

- Shell allows you to prompt for user input.
- Syntax:
  - `read varname1, [varname2, .. ]`

    example

```
echo “Enter your name: \c”
read my_name
```

3. Bash Control Structures

- if-then-else
- case
- loops
  - for
  - while
  - until
  - select

3.1 Simple if Statement

General Syntax:

```
if command
then
    statements
fi
```

statements are executed only if **command** succeeds, i.e. has return status “0”

3.2 if-then–else Statement

General syntax:

```
if [ condition ];
then
    statements-1
else
    statements-2
fi
```

- executes statements-1 if condition is true
- executes statements-2 if condition is false

3.3 If-then-elif-then-else Statement

General syntax:

```
if [ condition ]; then
  statements
elif [ condition ]; then
    statement
else
    statements
fi
```

- The word **elif** stands for “else if”
- It is part of the if statement and cannot be used by itself

3.4 Relational Operators

Meaning	Numeric	String
Greater than	-gt	-
Greater than or equal	-ge	-
Less than	-lt	-
Less than or equal	-le	-
Equal	-eq	= or ==
Not equal	-ne	!=

Meaning	Numeric	String
str1 is less than str2	-	str1 < str2
str1 is greater str2	-	str1 > str2
String length is greater than zero	-	-n str
String length is zero	-	-z str

3.5 File Testing

Options	Meaning
-d	True if file is a directory
-f	True if file is a ordinary file
-r	True if file is readable
-w	True if file is writable
-x	True if file is executable
-s	True if length of file is non - zero

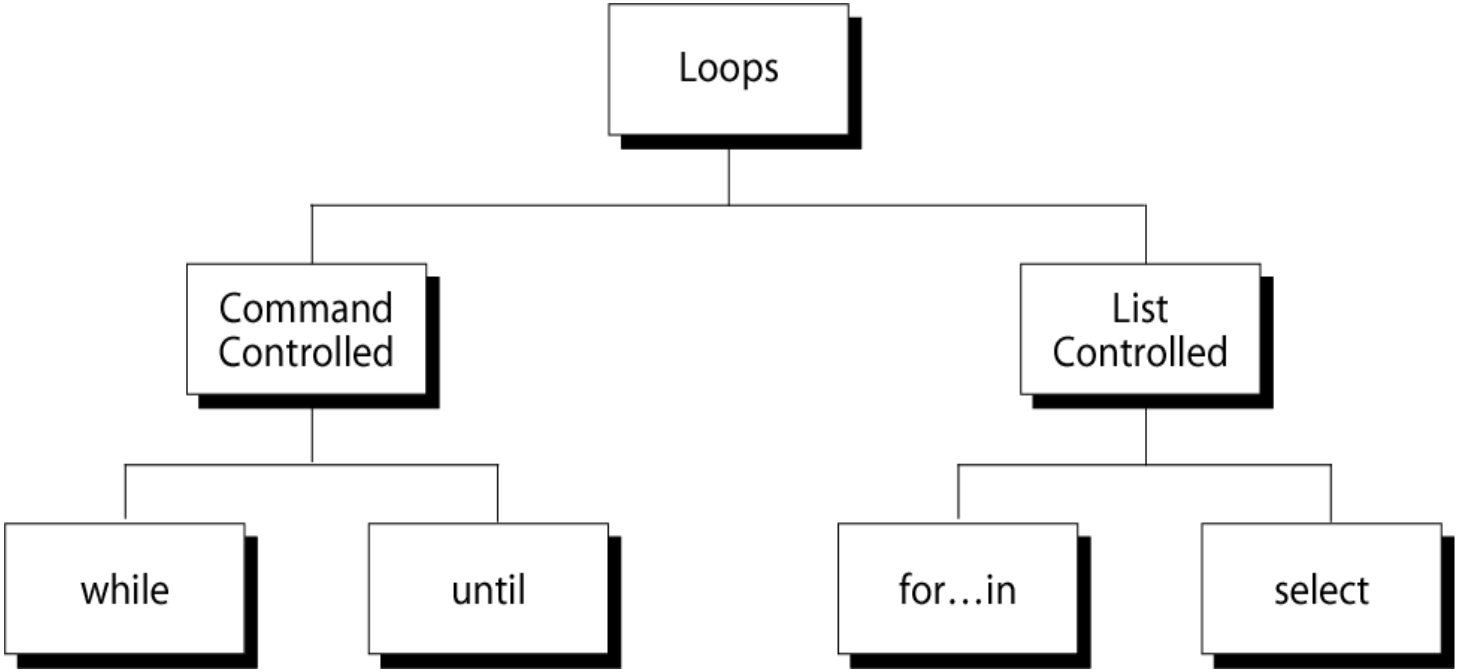
3.6 case Statement

- use the case statement for a decision that is based on multiple choices

General syntax:

```
case word in
pattern1) command-list1
;;
pattern2) command-list2
;;
patternN) command-listN
;;
esac
```

4. Repetition Constructs



4.1 while Loop

- To execute commands in “command-list” as long as “expression” evaluates to true

General syntax:

```
while [ expression ]

do

    command-list

done
```

4.2 The until Loop

- To execute commands in “command-list” as long as “expression” evaluates to false

General syntax:

```
until [ expression ]
do
    command-list
done
```

4.3 The for Loop

- To execute commands as many times as the number of words in the “argument-list”

General Syntax:

```
for variable in argument-list

do

    commands

done
```

4.4 select Command

- Constructs simple menu from word list
- Allows user to enter a number instead of a word
- User enters sequence number corresponding to the word

General Syntax:

```
select WORD in LIST
do
    RESPECTIVE-COMMANDS
done
```

Loops until end of input, i.e. ^d (or ^c)

4.5 break and continue

- Interrupt for, while or until loop
- The break statement
  - transfer control to the statement AFTER the done statement
  - terminate execution of the loop
- The continue statement
  - transfer control to the statement TO the done statement
  - skip the test statements for the current iteration
  - continues execution of the loop

5. Shell Functions

- A shell function is similar to a shell script
  - stores a series of commands for execution later
  - shell stores functions in memory
  - shell executes a shell function in the same shell that called it
- Where to define
  - In .profile
  - In your script
  - Or on the command line

General Syntax:

```
function-name () {
   statements
}
```

- must be defined before they can be referenced
- usually placed at the beginning of the script

- Remove a function
  - Use unset built-in

5.1 Function Parameters

- Need not be declared
- Arguments provided via function call are accessible inside function as $1, $2, $3, ...

  $# reflects number of parameters

  $0 still contains name of script (not name of function)

5.2 Local Variables in Function

- Variables defined within functions are global, i.e. their values are known throughout the entire shell program
- Keyword “local” inside a function definition makes referenced variables “local” to that function

6. Debugging Shell Programs

- Debugging is troubleshooting errors that may occur during the execution of a program/script
- The following two commands can help you debug a bash shell script:
  - echo - use explicit output statements to trace execution
  - set

6.1 Debugging Using “set”

- The “**set**” command is a shell built-in command has options to allow flow of execution
  - **–v** option prints each line as it is read
  - **–x** option displays the command and its arguments
  - **–n** checks for syntax errors

- options can turned on or off
  - To turn *on* the option: `set -xv`
  - To turn *off* the options: `set +xv`
- Options can also be set via she-bang line

```
#! /bin/bash -xv
```

7. Applications

- You can write a script which installs prequisites
- To kill or start multiple applications together
- To observe large database of files and find some patterns out of it
- So in general to automate the process and the list goes on

8. Limitations

- Inadvertent typing errors such as **`rm -rf * /`** (instead of the intended **`rm -rf */`**)
- need to launch a new process for almost every shell command executed, slow execution speed

References

- Beginning the Shell Scripting


FEEDBACK

Large enough to Deliver, Small enough to Care

Global Village
IT SEZ
Bangalore

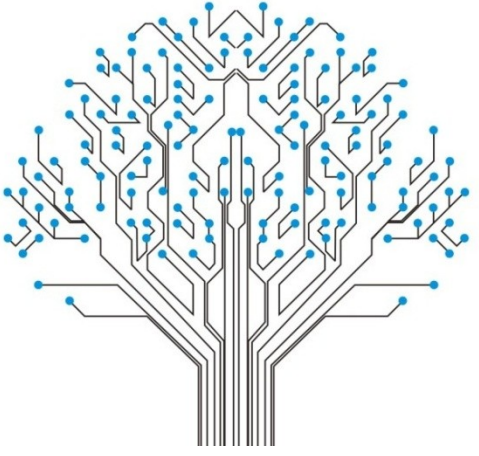
South Main Street
Milpitas
California

Raheja Mindspace
IT Park
Hyderabad

www.globaledgesoft.com

GLOBAL EDGE
Intelligence Of Things™

Thank you



Fairness

Learning

Responsibility

Innovation

Respect