Essential Shell Programming

Part I

What we will be learning

- What is Shell Programming
- Need for Shell Programming
- Shell Programming Variants
- Writing some Scripts

Basics

Definition:

Shell is an agency that sits between the user and the UNIX system.

Description:

- Understands all user directives and carries them out.
- Processes the commands issued by the user.
- Type of shell called Bourne shell.

What is Shell Programming

- Grouping a set commands
- Programming constructs used

Need for Shell Programming

- To execute a set of commands regularly
- Typing every time every command is laborious & time consuming
- To have control on the sequence of commands to be executed based previous results

Shell Scripts/Shell Programs

- Group of commands have to be executed regularly
- Stored in a file
- File itself executed as a shell script or a shell program by the user.
- A shell program runs in interpretive mode.
- Shell scripts are executed in a separate child shell process which may or may not be same as the login shell.

Shell Scripts

Example: script.sh

```
#! /bin/sh
# script.sh: Sample Shell Script
echo “Welcome to Shell Programming”
echo “Today’s date : `date`”
echo “This months calendar:”
cal `date “+%m 20%y”`
echo “My Shell :$ SHELL”
```

This month’s calendar.

Shell Scripts

To run the script we need to first make it executable. This is achieved by using the chmod command as shown below:

```
$ chmod +x script.sh
```

Then invoke the script name as:

```
$ script.sh
```

Shell Scripts

Explicitly spawn a child with script name as argument:

```
sh script.sh
```

Note: Here the script neither requires a executable permission nor an interpreter line.

Read: Making Scripts Interactive

- Shell's internal tool for making scripts interactive
- Used with one or more variables.
- Inputs supplied with the standard input are read into these variables.

Ex: *read name*

causes the script to pause at that point to take input from the keyboard.

Read: Making Scripts Interactive

Example: A shell script that uses read to take a search string and filename from the terminal.

```
#! /bin/sh
# emp1.sh: Interactive version, uses read to accept two
# inputs
        echo “Enter the pattern to be searched: \c”
        # No newline
        read pname
```

Read: Making Scripts Interactive

```
echo “Enter the file to be used: \c”
        read fname
        echo “Searching for pattern $pname from the
file $fname”
        grep $pname $fname
        echo “Selected records shown above”
```

Read: Making Scripts Interactive

Output:

```
$ emp1.sh
Enter the pattern to be searched : director
Enter the file to be used: emp.lst
Searching for pattern director from the file emp.lst
9876  Jai      Director Productions
2356 Rohit   Director  Sales
Selected records shown above
```

Read: Making Scripts Interactive

Output:

```
$ emp1.sh
Enter the pattern to be searched : director
Enter the file to be used: emp.lst
Searching for pattern director from the file emp.lst
9876  Jai      Director Productions
2356 Rohit  Director  Sales
Selected records shown above
```

Using Command Line Arguments

- Shell scripts accept arguments from the command line.
- Run non interactively
- Arguments are assigned to special shell variables (positional parameters).
- Represented by $1, $2, etc;

Using Command Line Arguments

Shell parameter	Significance
$1, $2...	Positional parameters representing command line arguments
$ #	No. of arguments specified in command line
$ 0	Name of the executed command
$ *	Complete set of positional parameters as a single string
"$ @"	Each quoted string treated as separate argument
$?	Exit status of last command
$$	Pid of the current shell
$!	PID of the last background job.

Using Command Line Arguments

```
#! /bin/sh
                    echo "Program Name : $0"
                    echo "No of Arguments : $#"
                    echo "Arguments are : $*"
$ chmod +x 2.sh
$ 2.sh A B C
o/p→        Program Name : 2.sh
            No of Arguments : 3
            Arguments are : A B C
```

Conclusion

In this session we have learnt

- Grouping of commands using the concept of shell scripts.
- Application of shell programming
- Providing information to the script interactively as well as through command line.