

Experiment No- 07

Aim: To Execute the basic shell scripts

RollNo.	17
Name	Manav Jawrani
Class	D10A
Subject	Unix Lab
Lab Outcome	LO4: To study shell, types of shell, variables and operators.
Date of Performance/ Submission	02/3/2022 - 10/3/2022

AIM : To Execute the basic shell scripts.

THEORY:

A shell script is a computer program designed to be run by the Unix/Linux shell which could be one of the following:

1. The Bourne Shell
2. The C Shell
3. The Korn Shell
4. The GNU Bourne-Again Shell

A shell is a command-line interpreter and typical operations performed by shell scripts include file manipulation, program execution, and printing text.

Shell scripts have several required constructs that tell the shell environment what to do and when to do it. Of course, most scripts are more complex than the above one.

The shell is, after all, a real programming language, complete with variables, control structures, and so forth. No matter how complicated a script gets, it is still just a list of commands executed sequentially.

A Shell provides you with an interface to the Unix system. It gathers input from you and executes programs based on that input. When a program finish executing, it displays that program's output.

Shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of a shell, just as there are different flavours of operating systems. Each shell has its own set of recognized commands and functions.

Shell Types

In Unix, there are two major types of shells –

- Bourne shell** – If you are using a Bourne-type shell, the \$ character is the default prompt.

- C shell** – If you are using a C-type shell, the % character is the default prompt.

The Bourne Shell has the following subcategories –

- **Bourne shell (sh)** - The Bourne shell is the original UNIX shell (command execution program, often called a command interpreter). Named for its developer, Stephen Bourne, the Bourne shell is also known by its program name, sh. The shell prompt (character displayed to indicate readiness for input) used is the \$ symbol. The Bourne shell family includes the Bourne, Kornshell, bash, and zsh shells.

- **Korn shell (ksh)** -The Korn shell belongs to the most common category, known as

character-based user interfaces. These interfaces accept lines of textual commands that the user types in; they usually produce text-based output.

- **Bourne Again shell (bash)** - Bourne again shell (Bash) is a free Unix shell that can be used in place of the Bourne shell. Bash is basically a command processor that typically runs in a text window, allowing the user to type commands that cause actions. It can read commands from a file, called a script.

- **POSIX shell (sh)** - POSIX is an acronym for “Portable Operating System Interface”. POSIX defines the application programming interface (API), along with Unix command line shells and utility interfaces. The POSIX standard is designed to be used by both application programmers and system administrators.

The different C-type shells follow –

1. **C shell (csh)**
2. **TENEX/TOPS C shell (tcsh)**

Variables:

A variable is nothing more than a pointer to the actual data. The shell enables you to create, assign, and delete variables.

The name of a variable can contain only letters (a to z or A to Z), numbers (0 to 9) or the underscore character (_).

• Arithmetic Operators

The following arithmetic operators are supported by Bourne Shell.

Operator	Description	Example
+ (Addition)	Adds values on either side of the operator	`expr \$a + \$b`

- (Subtraction)	Subtracts right hand operand from left hand operand	`expr \$a - \$b`
* (Multiplication)	Multiplies values on either side of the operator	`expr \$a * \$b`
/ (Division)	Divides left hand operand by right hand operand	`expr \$b /\$a`
% (Modulus)	Divides left hand operand by right hand operand and returns remainder	`expr \$b % \$a`
= (Assignment)	Assigns right operand in left operand	a = \$b would assign value of b into a
== (Equality)	Compares two numbers,if both are same then returns true	[\$a == \$b] would return false
!= (Not Equality)	Compares two numbers,if both are different then returns true	[\$a != \$b] would return true

It is very important to understand that all the conditional expressions should be inside square braces with spaces around them, for example [**\$a = = \$b**] is correct whereas, [**\$a= =\$b**] is incorrect.

All the arithmetical calculations are done using long integers.

Scripts :

(i) Write a shell script to perform arithmetic

operations.

Script:

`#!/bin/sh``echo Enter first number:``read a``echo Enter second number:``read b``echo "First number = $a"``echo "Second number = $b"``val=`expr $a + $b```echo "Addition Of Two Numbers: a + b : $val"``val=`expr $a - $b```echo "a - b : $val"``val=`expr $a * $b```echo "a * b : $val"``val=`expr $b / $a```echo "b / a : $val"``val=`expr $b % $a```echo "b % a : $val"`

Input:

First number = a = 17

Second number = b = 27

]

Output:

```
test1@510-13-hp280:~/Desktop$ chmod +x Arithmetic.sh
test1@510-13-hp280:~/Desktop$ ./Arithmetic.sh
Enter first number:
17
Enter second number:
27
First number = 17
Second number = 27
Addition Of Two Numbers: a + b : 44
a - b : -10
a * b : 459
b / a : 1
b % a : 10
test1@510-13-hp280:~/Desktop$
```

ii) Write a shell script to calculate simple interest.

Script:

```
#!/bin/bash
```

```
echo Enter the principle value[in Rupees]:
```

```
read p
```

```
echo Enter the time period value[in Years]:
```

```
read n
```

```
echo Enter the rate of interest[in Percentage]:
```

```
read r
```

```
val=`expr $p \* $n \* $r / 100`
```

```
echo "Simple Interest is = $val"
```

Input:

Principal value (p) = 1,80,000

Time period (n) = 5 years

Rate of interest (r) = 12%

Output:

```
test1@510-13-hp280:~/Desktop$ ./SimpleInterest.sh
Enter the principle value[in Rupees]:
180000
Enter the time period value[in Years]:
5
Enter the rate of interest[in Percentage]:
12
Simple Interest is = 108000
test1@510-13-hp280:~/Desktop$
```

(iii) Write a shell script to determine largest among

three integer numbers.

Script:

```
#!/bin/bash
```

```
echo Enter first number:
```

```
read num1
```

```
echo Enter second number:
```

```
read num2
```

```
echo Enter third number:
```

```
read num3
```

```
if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]
```

```
then
```

```
    echo Greatest Number is : $num1
```

```
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]
```

```
then
```

```
    echo Greatest Number is : $num2
```

```
else
```

```
    echo Greatest Number is : $num3
```

```
fi
```

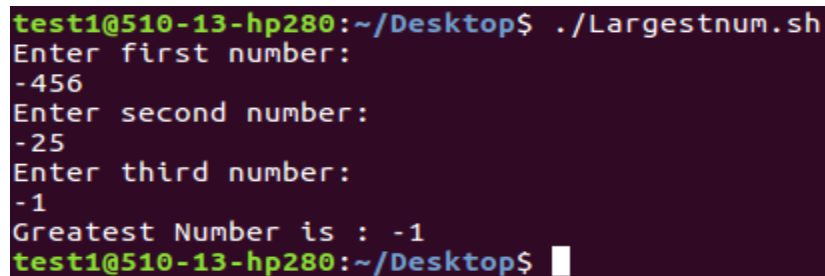
Input:

Number 1 (num1) = -456

Number 2 (num2) = -25

Number 3 (num3) = -1

Output:



```
test1@510-13-hp280:~/Desktop$ ./Largestnum.sh
Enter first number:
-456
Enter second number:
-25
Enter third number:
-1
Greatest Number is : -1
test1@510-13-hp280:~/Desktop$
```

(iv) Write a shell script to determine a given year is

leap year or not.

Script:

```
#!/bin/bash
```

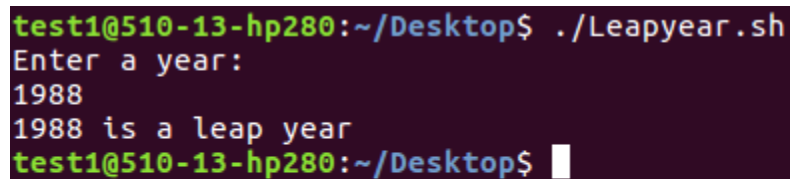
```
echo "Enter a year:"
read year_checker
if [ `expr $year_checker % 4` -eq 0 ]
then
    echo "$year_checker is a leap year"
else
    echo "$year_checker is not a leap year"
fi
```

Input:

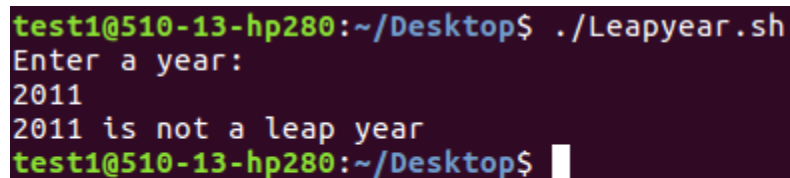
Leap year - 1988

Not a leap year - 2011

Output:



```
test1@510-13-hp280:~/Desktop$ ./Leapyear.sh
Enter a year:
1988
1988 is a leap year
test1@510-13-hp280:~/Desktop$
```



```
test1@510-13-hp280:~/Desktop$ ./Leapyear.sh
Enter a year:
2011
2011 is not a leap year
test1@510-13-hp280:~/Desktop$
```

Conclusion:

We have understood what are shell scripts and wrote all the shell scripts for the given problem statement.