Department of
Computer Science and Engineering

Title: Shell Scripting- II

Operating System Lab

CSE 310

Green University of Bangladesh

1 Objective(s)

- To gather knowledge of shell program.

2 Shell Scripting

A shell script is a computer program designed to be run by the Unix shell, a command-line interpreter. The various dialects of shell scripts are considered to be scripting languages. Typical operations performed by shell scripts include file manipulation, program execution, and printing text. A script which sets up the environment, runs the program, and does any necessary cleanup, logging, etc. is called a wrapper.

2.1 Loop

Most languages have the concept of loops: If we want to repeat a task twenty times, we don't want to have to type in the code twenty times, with maybe a slight change each time. As a result, we have **for, until and while** loops in the Bourne shell. This is somewhat fewer features than other languages, but nobody claimed that shell programming has the power of C.

2.1.1 For Loop Implementation in shell

```
1
2
3  #!/bin/bash
4  for i in 1 2 3 4 5
5  do
6     echo "Welcome $i times"
7  done
8
9  -------------------------
10
11 #!/bin/bash
12 for i in {1..5}
13 do
14    echo "Welcome $i times"
15 done
16
17 -------------------------
18
19  #!/bin/bash
20 for i in {0..10..2}
21   do
22      echo "Welcome $i times"
23  done
24
25 -------------------------
26
27  #!/bin/bash
28 for i in $(seq 1 2 20)
29 do
30    echo "Welcome $i times"
31 done
32
33 -------------------------
34
35 #!/bin/bash
36 for (( c=1; c<=5; c++ ))
37 do
```

```
    echo "Welcome $c times"
done
```

2.1.2 Lab Task (Please implement yourself and show the output to the instructor)

1. Write a Shell program to find the sum of odd and even numbers from a set of numbers.
2. Write a Shell program to find the smallest number from a set of numbers.
3. Write a Shell program to find the sum of all numbers between 50 and 100, which are divisible by 3 and not divisible by 5.
4. Write a Shell program to find the second highest number from a set of numbers.
5. Write a Shell program to find the factorial of a number using for loop.
6. Write a Shell program to generate Fibonacci series.

2.1.3 While Loop Implementation in shell

```


#!/bin/bash
# set n to 1
n=1

# continue until $n equals 5
while [ $n -le 5 ]
do
  echo "Welcome $n times."
  n=$(( n+1 ))    # increments $n
done
--------------------------

#!/bin/bash
n=1
while (( $n <= 5 ))
do
  echo "Welcome $n times."
  n=$(( n+1 ))
done
--------------------------

#!/bin/sh
INPUT_STRING=hello
while [ "$INPUT_STRING" != "bye" ]
do
  echo "Please type something in (bye to quit)"
  read INPUT_STRING
  echo "You typed: $INPUT_STRING"
done
```

2.1.4 Lab Task (Please implement yourself and show the output to the instructor)

1. Write a Shell program to find the smallest digit from a number.
2. Write a Shell program to find the second largest digit from a number.
3. Write a Shell program to find the sum of digits of a number.
4. Write a Shell program to check the given integer is Armstrong number or not.

2.1.5 Until Loop

The until loop is used to execute a given set of commands as long as the given condition evaluates to false.

2.1.6 Implementation in shell

```
1
2
3  #!/bin/bash
4
5  counter=0
6
7  until [ $counter -gt 5 ]
8  do
9    echo Counter: $counter
10   ((counter++))
11 done
```

2.1.7 Lab Task (Please implement yourself and show the output to the instructor)

1. Write a Shell program to find the sum of odd digits and even digits from a number.
2. Write a Shell program to find the largest digit of a number.

2.2 Functions

Functions enable you to break down the overall functionality of a script into smaller, logical subsections, which can then be called upon to perform their individual tasks when needed.
Using functions to perform repetitive tasks is an excellent way to create code reuse. This is an important part of modern object-oriented programming principles.

2.2.1 Implementation in shell

```
1
2  #!/bin/sh
3
4  # Define your function here
5  Hello () {
6     echo "Hello World"
7  }
8
9  # Invoke your function
10 Hello
```

```
11
12  ------------------------------
13  #!/bin/sh
14  #Pass Parameters to a Function
15  # Define your function here
16  Hello () {
17     echo "Hello World $1 $2"
18  }
19
20  # Invoke your function
21  Hello Zara Ali
22
23  ------------------------------
24
25  #!/bin/sh
26  #Nested Functions (Recursive function)
27  # Calling one function from another
28  number_one () {
29     echo "This is the first function speaking..."
30     number_two
31  }
32
33  number_two () {
34     echo "This is now the second function speaking..."
35  }
36
37  # Calling function one.
38  number_one
39
40  ------------------------------
41  # A simple example of a factorial function
42  #!/bin/sh
43
44  factorial()
45  {
46    if [ "$1" -gt "1" ]; then
47      i=`expr $1 - 1`
48      j=`factorial $i`
49      k=`expr $1 \* $j`
50      echo $k
51    else
52      echo 1
53    fi
54  }
55
56
57  while :
58  do
59    echo "Enter a number:"
60    read x
61    factorial $x
62  done
```

2.2.2 Lab Task (Please implement yourself and show the output to the instructor)

1. Write a Shell program to find the largest number between two numbers using function.
2. Write a Shell program to find the sum of the numbers passed as parameters.

2.3 Arrays

An array is a systematic arrangement of the same type of data. But in Shell script Array is a variable which contains multiple values may be of same type or different type since by default in shell script everything is treated as a string. An array is zero-based ie indexing start with 0.

2.3.1 Implementation in shell

```
#! /bin/bash
# To declare static Array
arr=(prakhar ankit 1 rishabh manish abhinav)

# To print all elements of array
echo ${arr[@]}          # prakhar ankit 1 rishabh manish abhinav
echo ${arr[*]}          # prakhar ankit 1 rishabh manish abhinav
echo ${arr[@]:0}     # prakhar ankit 1 rishabh manish abhinav
echo ${arr[*]:0}     # prakhar ankit 1 rishabh manish abhinav

# To print first element
echo ${arr[0]}          # prakhar
echo ${arr}              # prakhar

# To print particular element
echo ${arr[3]}          # rishabh
echo ${arr[1]}          # ankit

# To print elements from a particular index
echo ${arr[@]:0}     # prakhar ankit 1 rishabh manish abhinav
echo ${arr[@]:1}     # ankit 1 rishabh manish abhinav
echo ${arr[@]:2}     # 1 rishabh manish abhinav
echo ${arr[0]:1}     # rakhar

# To print elements in range
echo ${arr[@]:1:4}     # ankit 1 rishabh manish
echo ${arr[@]:2:3}     # 1 rishabh manish
echo ${arr[0]:1:3}     # rak

# Length of Particular element
echo ${#arr[0]}          # 7
echo ${#arr}         # 7

# Size of an Array
echo ${#arr[@]}          # 6
echo ${#arr[*]}          # 6

# Search in Array
echo ${arr[@]/*[aA]*/}     # 1

# Replacing Substring Temporary
echo ${arr[@]//a/A}          # prAkhAr Ankit 1 rishAbh mAnish AbhinAv
echo ${arr[@]}              # prakhar ankit 1 rishabh manish abhinav
echo ${arr[0]//r/R}          # pRakhaR
```

2.3.2 Input/Output

Output of the program is given below.

```
prakhar ankit 1 rishabh manish abhinav
prakhar ankit 1 rishabh manish abhinav
prakhar ankit 1 rishabh manish abhinav
prakhar ankit 1 rishabh manish abhinav
prakhar
prakhar
rishabh
ankit
prakhar ankit 1 rishabh manish abhinav
ankit 1 rishabh manish abhinav
1 rishabh manish abhinav
rakhar
ankit 1 rishabh manish
1 rishabh manish
rak
7
7
6
6
1
prAkhAr Ankit 1 rishAbh mAnish AbhinAv
prakhar ankit 1 rishabh manish abhinav
pRakhaR
```

2.3.3 Implementation in shell

```
1
2  # !/bin/bash
3  #By Using while-loop
4  # To declare static Array
5  arr=(1 12 31 4 5)
6  i=0
7
8  # Loop upto size of array
9  # starting from index, i=0
10 while [ $i -lt ${#arr[@]} ]
11 do
12     # To print index, ith
13     # element
14     echo ${arr[$i]}
15
16     # Increment the i = i + 1
17     i=`expr $i + 1`
18 done
```

2.3.4 Input/Output

Output of the program is given below.

```
1
12
31
4
5
```

2.3.5 Implementation in shell

```
1
2  # !/bin/bash
3  #By Using for-loop
4  # To declare static Array
5  arr=(1 2 3 4 5)
6
7  # loops iterate through a
8  # set of values until the
9  # list (arr) is exhausted
10 for i in "${arr[@]}"
11 do
12     # access each element
13     # as $i
14     echo $i
15 done
```

2.3.6 Input/Output

Output of the program is given below.

```
1
2
3
4
5
```

2.3.7 Lab Task (Please implement yourself and show the output to the instructor)

1. Write a Shell program to find the sum of odd and even numbers.
2. Write a Shell program to find the average of n numbers.
3. Write a Shell Program to find the largest element of an array.
4. Write a Shell Program to find the smallest element of an array.

3 Discussion & Conclusion

Based on the focused objective(s) to understand about the shell program, the additional lab exercise made me more confident towards the fulfilment of the objectives(s).

4 Lab Exercise (Submit as a report)

- Write a shell program to display odd position numbers (using For loop).

  Sample Input:
  Enter 7-digit number: 5867458

  Output:
  5
  6
  4
  8

- Write a Shell program using while loop:

  Sample Input:
  Enter the number: 148541547854

  Output:
  1 = 2 times
  4 = 4 times
  8 = 2 times
  5 = 3 times
  7 = 1 times

- Write a Shell program to find the 2nd highest and 3rd highest numbers from a set of numbers and sum of them using array.

  Sample Input:
  Enter the number of elements: 5
  Enter the number: 10
  Enter the number: 21
  Enter the number: 30
  Enter the number: 17
  Enter the number: 5

  Output:
  The sum of first and last element is: (21+17) = 38

- Write a Shell program to find the factorial of two different numbers and sum of the numbers using function.

  Sample Input:
  Factorial of 5 is 120
  Factorial of 6 is 720

  Output:
  120 + 720 = 840

- Write a Shell program to find total number of alphabets, digits or special characters in a string.

  Sample Input:
  Today is 12 November.

  Output:
  Alphabets = 15
  Digits = 2
  Special characters = 4

5 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.