Department of
Computer Science and Engineering

Title: Shell Scripting- I

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 Arithmetic

Doing arithmetic operations in bash is not similar to other standard programming languages. One of the limitations of bash is that it can't handle floating point or double numbers like other scripting languages.

2.1.1 Let

let is a builtin function of Bash that allows us to do simple arithmetic. It follows the basic format:
let <arithmetic expression>

2.1.2 Implementation in shell

```bash

#!/bin/bash
# Basic arithmetic using let

let a=5+4
echo $a # 9

let "a = 5 + 4"
echo $a # 9

let a++
echo $a # 10

let "a = 4 * 5"
echo $a # 20

let "a = $1 + 30"
echo $a # 45
```

2.1.3 Input/Output

Output of the program is given below.

```
./let_example.sh 15
9
9
10
20
45
```

2.1.4 Expr

expr is similar to let except instead of saving the result to a variable it instead prints the answer. Unlike let you don't need to enclose the expression in quotes. You also must have spaces between the items of the expression. It is also common to use expr within command substitution to save the output to a variable.

expr item1 operator item2

2.1.5 Implementation in shell

```
#!/bin/bash
# Basic arithmetic using expr

expr 5 + 4 # 9

expr "5 + 4" # 5 + 4

expr 5+4 # 5+4

expr 5 \* $1 # 60

expr 11 % 2 # 1

a=$( expr 10 - 3 )
echo $a # 7
```

2.1.6 Input/Output

Output of the program is given below.

```
./expr_example.sh 12
9
5 + 4
5+4
60
1
7
```

2.1.7 Double Parentheses

In the section on Variables we saw that we could save the output of a command easily to a variable. It turns out that this mechanism is also able to do basic arithmetic for us if we tweak the syntax a little. We do so by using double brackets like so:

$((expression))

2.1.8 Implementation in shell

```
#!/bin/bash
# Basic arithmetic using double parentheses

a=$(( 4 + 5 ))
echo $a # 9

a=$((3+5))
echo $a # 8

b=$(( a + 3 ))
echo $b # 11
```

```

b=$(( $a + 4 ))
echo $b # 12

(( b++ ))
echo $b # 13

(( b += 3 ))
echo $b # 16

 a=$(( 4 * 5 ))
echo $a # 20
```

2.1.9 Input/Output

Output of the program is given below.

```
./expansion_example.sh
9
8
11
12
13
16
20
```

2.1.10 Floating-point arithmetic

In Bash shell, we can only perform integer arithmetic. If we want to perform arithmetic involving a floating point or fractional values, then we will need to use various other utilities, such as awk, bc, and similar.

Let's see an example of using the utility called bc:

2.1.11 Implementation in shell

```
#!/bin/bash
# Floating-point arithmetic
echo "scale = 5; 123.456789/345.345345" | bc # .35748

echo 'scale=4;20+5/2' | bc # 22.5000

--------------------------------
#Bash Shell Script to convert Celsius to Fahrenheit
#!/bin/bash
read -p "Enter degree celsius temperature: " celsius
fahrenheit=`echo "scale=4; $celsius*1.8 + 32" | bc`
echo "$celsius degree celsius is equal to $fahrenheit degree fahrenheit"

#37 degree celsius is equal to 98.6 degree fahrenheit
```

For using the bc utility, we need to configure a scale parameter. Scale is the number of significant digits to the right of the decimal point.

2.1.12 Length of a Variable

This isn't really arithmetic but it can be quite useful. If you want to find out the length of a variable (how many characters) you can do the following:

$#variable

2.1.13 Implementation in shell

```bash
#!/bin/bash
# Show the length of a variable.

a='Hello World'
echo ${#a} # 11

b=4953
echo ${#b} # 4
```

2.1.14 Input/Output

Output of the program is given below.

```
1. ./length_example.sh
2. 11
3. 4
```

2.1.15 Takes input from the user at run time

```bash
#!/bin/bash
# Take input from user and calculate sum.

read -p "Enter first number: " num1
read -p "Enter second number: " num2

sum=$(( $num1 + $num2 ))

echo "Sum is: $sum"
```

2.1.16 Input/Output

Output of the program is given below.

```
Enter first number: 12
Enter second number: 15
Sum is: 27
```

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

1. Write a Shell program to find the area and circumference of a circle.
2. Write a Shell program to find the roots of a quadratic equation.
3. Write a Shell program to Find out the Area and Perimeter of Rectangle.
4. Write a Shell program to Find the Perimeter of a Circle, Rectangle and Triangle

2.2 Basic String Operations

The shell allows some common string operations which can be very useful for script writing.

```
#       1234567890123456
STRING="this is a string"
echo ${#STRING}             # 16
```

```
#Find the numerical position in $STRING of any single character in $SUBSTRING
    that matches. Note that the 'expr' command is used in this case.
STRING="this is a string"
SUBSTRING="hat"
expr index "$STRING" "$SUBSTRING"     # 1 is the position of the first 't' in
    $STRING
```

```
#Extract substring of length $LEN from $STRING starting after position $POS.
    Note that first position is 0.
STRING="this is a string"
POS=1
LEN=3
echo ${STRING:$POS:$LEN}    # his
```

```
#! /bin/bash

var="Welcome to the geekstuff"

echo ${var:15} # geekstuff
echo ${var:15:4} # geek
```

```
#If :$LEN is omitted, extract substring from $POS to end of line
STRING="this is a string"
echo ${STRING:1}            # $STRING contents without leading character
echo ${STRING:12}           # ring
```

```
#Replace first occurrence of substring with replacement
STRING="to be or not to be"
echo ${STRING[@]/be/eat}         # to eat or not to be
```

```
#Replace all occurrences of substring
STRING="to be or not to be"
echo ${STRING[@]//be/eat}         # to eat or not to eat
```

```
#Delete all occurrences of substring (replace with empty string)
STRING="to be or not to be"
echo ${STRING[@]// not/}         # to be or to be
```

```
#Replace occurrence of substring if at the beginning of $STRING
STRING="to be or not to be"
echo ${STRING[@]/#to be/eat now}     # eat now or not to be
```

```
#Replace occurrence of substring if at the end of $STRING
STRING="to be or not to be"
echo ${STRING[@]/%be/eat}         # to be or not to eat
```

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

1. Write a shell Script to Concatenate Two Strings.

2.3 Conditional Statements

There are total 5 conditional statements which can be used in bash programming.

- if statement
- if-else statement
- if..elif..else..fi statement (Else If ladder)
- if..then..else..if..then..fi..fi..(Nested if)
- switch statement

2.3.1 Implementation in shell

```
#Initializing two variables
a=10
b=20

#Check whether they are equal
if [ $a == $b ]
then
    echo "a is equal to b"
fi

#Check whether they are not equal
if [ $a != $b ]
then
    echo "a is not equal to b"
fi
```

2.3.2 Input/Output

Output of the program is given below.

```
a is not equal to b
```

2.3.3 Implementation in shell

```
#!/bin/bash
# Basic if statement
if [ $1 -gt 100 ]
then
echo Hey that\'s a large number.

fi
date
```

2.3.4 Input/Output

Output of the program is given below.

```
./if_example.sh 15
Fri 2 Jul 15:38:39 2021 ./if_example.sh 150
Hey that's a large number.
Fri 2 Jul 15:38:39 2021
```

2.3.5 Implementation in shell

```sh
#Initializing two variables
a=20
b=20

if [ $a == $b ]
then
    #If they are equal then print this
    echo "a is equal to b"
else
    #else print this
    echo "a is not equal to b"
fi
```

2.3.6 Input/Output

Output of the program is given below.

```
a is not equal to b
```

2.3.7 Implementation in shell

```sh

#!/bin/sh

a=10
b=20

if [ $a == $b ]
then
   echo "a is equal to b"
elif [ $a -gt $b ]
then
   echo "a is greater than b"
elif [ $a -lt $b ]
then
   echo "a is less than b"
else
   echo "None of the condition met"
fi
```

2.3.8 Input/Output

Output of the program is given below.

```
a is less than b
```

2.3.9 Implementation in shell

```
CARS="bmw"

#Pass the variable in string
case "$CARS" in
    #case 1
    "mercedes") echo "Headquarters - Affalterbach, Germany" ;;

    #case 2
    "audi") echo "Headquarters - Ingolstadt, Germany" ;;

    #case 3
    "bmw") echo "Headquarters - Chennai, Tamil Nadu, India" ;;
esac
```

2.3.10 Input/Output

Output of the program is given below.

```
Headquarters - Chennai, Tamil Nadu, India.
```

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

1. Write a Shell program to find the largest among three numbers.
2. Write a Shell program to display student grades using switch case.

2.4 Compare Strings in Bash

2.4.1 Implementation in shell

```
#!/bin/bash

VAR1="Linuxize"
VAR2="Linuxize"

if [ "$VAR1" = "$VAR2" ]; then
    echo "Strings are equal."
else
    echo "Strings are not equal."
fi
```

2.4.2 Input/Output

Output of the program is given below.

```
Strings are equal.
```

2.4.3 Implementation in shell

```
#!/bin/bash

read -p "Enter first string: " VAR1
read -p "Enter second string: " VAR2

if [[ "$VAR1" == "$VAR2" ]]; then
    echo "Strings are equal."
else
    echo "Strings are not equal."
fi
```

2.4.4 Input/Output

Output of the program is given below.

```
Enter first string: Linuxize
Enter second string: Ubuntu
Strings are not equal.
```

2.4.5 Implementation in shell

```
#There are multiple ways to check if a string contains a substring.

#One approach is to use surround the substring with asterisk symbols * which
   means match all characters.
#!/bin/bash

VAR='GNU/Linux is an operating system'
if [[ $VAR == *"Linux"* ]]; then
  echo "It's there."
fi

------------------------------
#!/bin/bash

strval="Microsoft Internet Explorer"

if [[ $strval == *Internet* ]];
then
  echo "Partially Match"
else
  echo "No Match"
fi

if [[ $strval == *internet* ]];
then
  echo "Partially Match"
else
  echo "No Match"
fi
```

2.4.6 Implementation in shell

```bash
#Check if a String is Empty
#Quite often you will also need to check whether a variable is an empty string
    or not. You can do this by using the -n and -z operators.

#!/bin/bash

VAR=''
if [[ -z $VAR ]]; then
  echo "String is empty."
fi

#!/bin/bash

VAR='Linuxize'
if [[ -n $VAR ]]; then
  echo "String is not empty."
fi
```

2.4.7 Input/Output

Output of the program is given below.

```
String is empty.
String is not empty.
```

2.4.8 Implementation in shell

```bash


#!/bin/bash

VAR="Arch Linux"

case $VAR in

  "Arch Linux")
    echo -n "Linuxize matched"
    ;;

  Fedora | CentOS)
    echo -n "Red Hat"
    ;;
esac
```

2.4.9 Input/Output

Output of the program is given below.

```
Linuxize matched.
```

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 find the sum of odd and even numbers from a set of numbers.
- Write a Shell program to Check Triangle is Valid or Not

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.