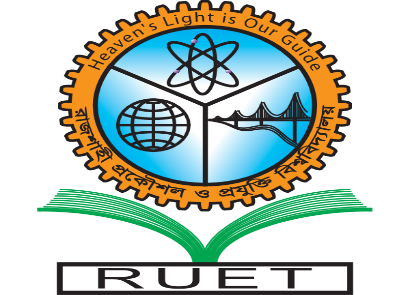
**Heaven’s Light is Our Guide**



**Computer Science And Engineering**

**Rajshahi University of Engineering and Technology**

**Course No: CSE3202**

**Course Title: Operating system**

**Date of Submission: 31-10-22**

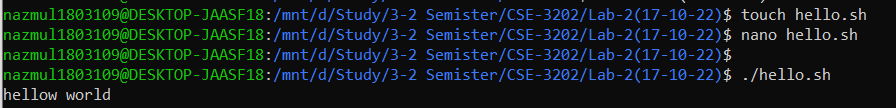
|  |  |
| --- | --- |
| **Submitted To** | **Submitted By** |
| **Mohiuddin Ahmed**  **Lecturer,**  **Department Of Computer Science And Engineering**  **Rajshahi University Of Engineering And Technology** | **Nazmul Haque**  **Roll: 1803109**  **Section: B**  **Department Of Computer Science And Engineering**  **Rajshahi University Of Engineering And Technology** |

**1)First Shell Code:**

**Code:**



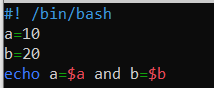
**Output:**



**Note:** First of all create hello.sh file using touch. Then open that file using nano and write code. Finally run that code using ./file\_name(such as ./hello.sh). Here, echo work as a printf(“”) function like c programming.

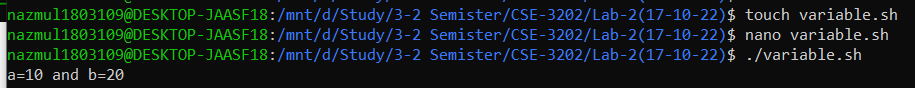
**2)Variables:**

**Code:**



**Note:** To assign the value of a variable we need to use ‘$’ sign at the front of that variable.

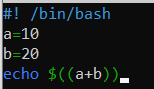
**Output:**



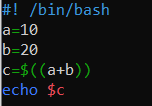
**3)Arithmetic Operation:**

**i) Summation Operation:**

**Code:**

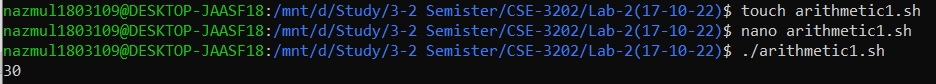


Or,



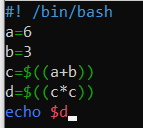
**Note:** We can use one of this two way.

**Output:**

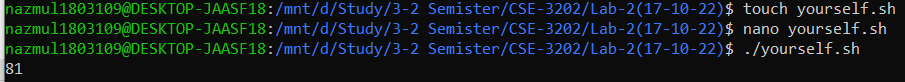


**ii)Try Yourself: (a+b)^2**

**Code:**

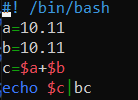


**Output:**



**iii) Sumation of floating point number:**

**Code:**



**Note:** Here use bc which is basic calculator liabrary.

**Output:**





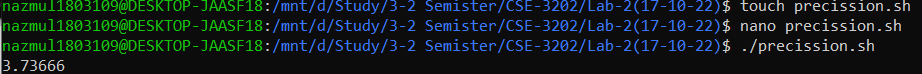
**iv) Precision of number:**

**Code:**



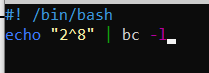
**Note:** if we want to 5 digit after decimal point then 5 should assign in scale.

**Output:**



**v) Power:**

**Code:**



#-l is used to invoke math library

**Output:**





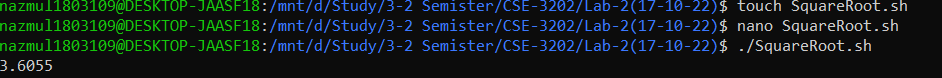
**vi) Square Root:**

**Code:**



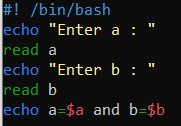
**Note:** to calculate square of a number need to basic calculator liabrary function.

**Output:**

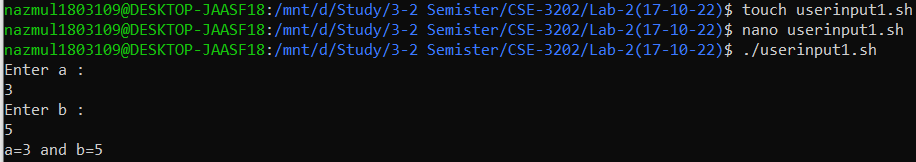


**4)Input from User**

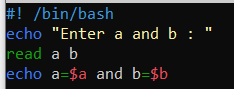
**Code:**



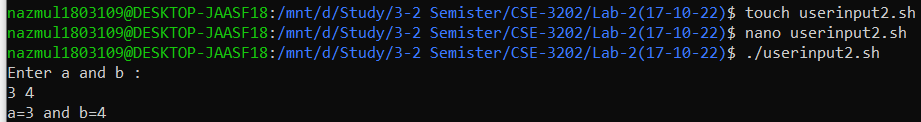
**Output:**



Or,

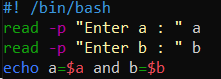


**Output:**



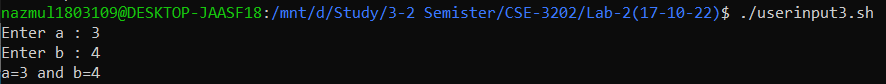
Or,

**Code:**



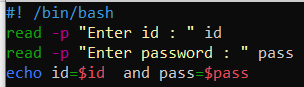
**Output:**



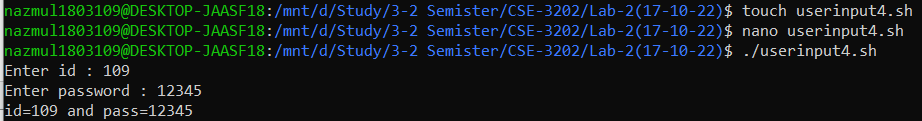


Or,

**Code:**



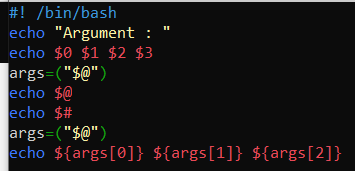
**Output:**



**Note:** we can use any of them way to take user input using read.

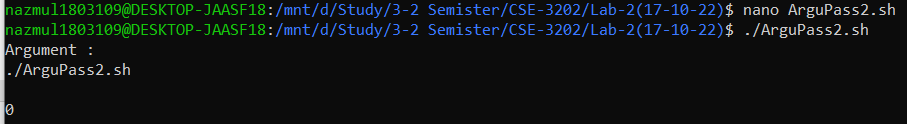
**5) Pass Argument during execution**

**Code:**

****

**Output:**

****



**6) Conditional Statement- If:**

**Syntax:**

|  |
| --- |
| if [ condition ]  then  #code to be executed if the condition is satisfied  else  #code to be executed if the condition is not satisfied  fi |
| if [ condition ] && [condition]  then  #code to be executed if the condition is satisfied  else  #code to be executed if the condition is not satisfied  fi |
| if [ condition ] **||** [condition]  then  #code to be executed if the condition is satisfied  else  #code to be executed if the condition is not satisfie  fi |

**Condition:**

* **-eq :** equals to

example: if [ $var -eq 0 ]

* **-ne :** not equals to

example: if [ $var -q ne 0 ]

* **-gt Or > :** Greater than

example: if [ $var -gt 0 ]

if [ $var > 0 ]

* **-lt Or <** : Less than

example: if [ $var -gt 0 ]

if [ $var > 0 ]

* **-ge Or >= :** Greater than equals to

example: if [ $var -ge 10 ]

if [ $var >= 10 ]

* **-le Or <= :** Greater than equals to

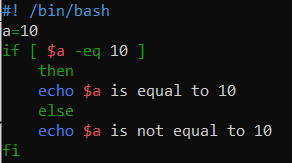
example: if [ $var -le 10 ]

if [ $var <= 10 ]

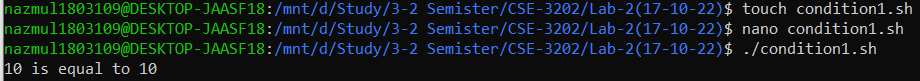
using this above syntax solve some simple problem below.

**i)use of Equal operator:**

**Code:**

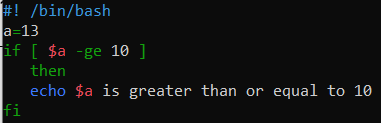


**Output:**

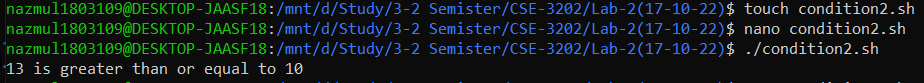


**ii)Use of greater than or equal operator:**

**Code:**



**Output:**



**iii) Conditional for String:**

* **== :** equals to

example: if [ $str == “value” ]

* **!= :** not equals to

example: if [ $str != “value” ]

* **< :** is less than in ASCII value

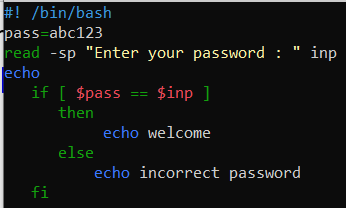
example: if [ $var -q ne 0 ]

* **> :** is greater than in ASCII value

example: if [ $var -q ne 0 ]

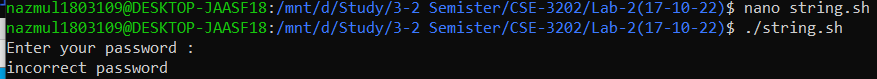
**example:**

**Code:**



**Output:**





**7) Loop Statement:**

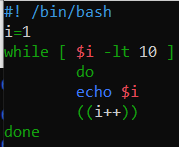
* **While:**

**Syntax:**

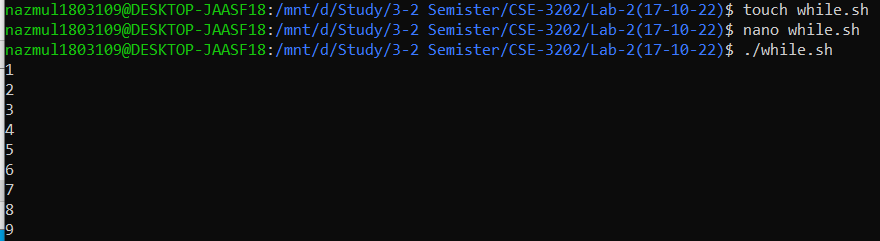
|  |
| --- |
| while [ condition ]  do  #code to be executed as long as the condition is satisfied  done |

**Ex.**

**Code:**

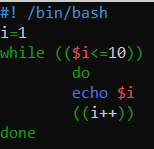


**Output:**

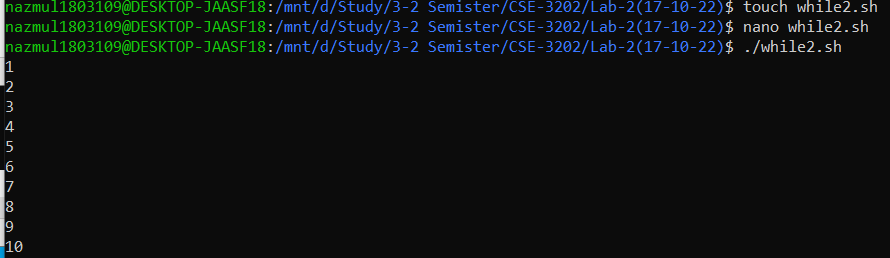


Or,

**Code:**



**Output:**



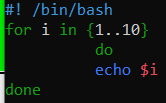
* **For :**

**Syntax:**

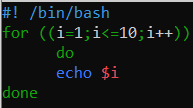
|  |
| --- |
| for variable in {range\_start..range\_end}  #code to be executed as long as the condition is satisfied  done |
| for ((start; condition; stepsize))  do  #code to be executed as long as the condition is satisfied  done |

Ex.

**Code:**

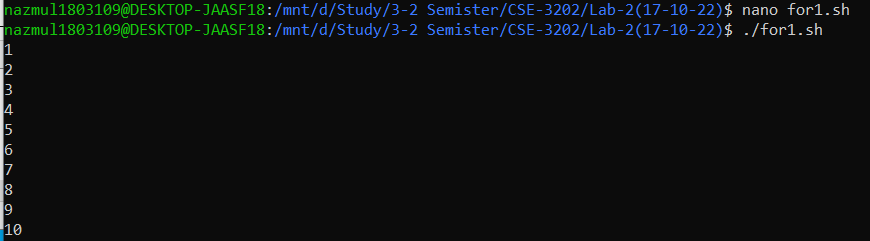


**Or,**



**Output:**





**8) Array**

1. **Indirect Declaration**

**ARRAYNAME[INDEXNR]=value**

1. **Explicit Declaration**

**declare -a ARRAYNAME**

1. **Compound Assignment**

**ARRAYNAME=(value1 value2 .... valueN)**

**Or**

**ARRAYNAME=([1]=10 [2]=20 [3]=30)**

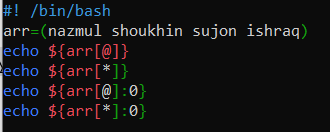
* **To print all the value of an array:**

echo ${ARRAYNAME[\*]}

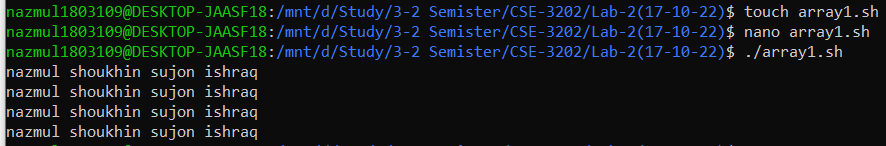
[@] & [\*] means All elements of Array.

Run this program:

**Code:**



**Output:**

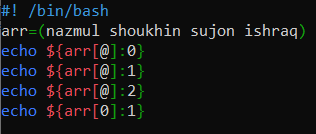


* **To print elements from a particular index**

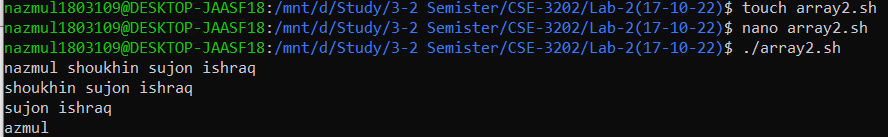
echo ${ARRAYNAME[WHICH\_ELEMENT]:STARTING\_INDEX}

Run the following code:

**Code:**



Output:



**Discussion:**

All of the above code run successfully.But I faced a problem for using syntax, missing space is the main caused for an error. In shell coding for loop is almost similar like other high level language(such as c,c++,python). We can relate shell coding with other high level language partially.