**Bahria University, Lahore Campus**

Department of Computer Sciences

Lab Journal 04

**(spring 2024)**

|  |  |  |
| --- | --- | --- |
| Course: | **Operating System Lab** | Date: 03-14-2024 |
| Course Code: | CSL-320 | Max Marks: 20 |
| Faculty’s Name: | ABDULLAH |  |

Name: Muhammad Hammad Enroll No: 03-134221-024

**Objective(s):**

To study loops, arrays, strings, file testing, and positional parameters.

**Tool(s) used:**

Ubuntu, VIM Editor

**Lab Tasks :  
Task 1:** Write a program to display the numbers from 10 to 20 in reverse order using for loop.   
**Task 2:** Write the output of the programs of Array.   
**Task 3:** Write the output of the File Testing Program.  
**Task 4:** Write a shell program to compare the two strings, whether the strings are equal or not.

**Lab Grading Sheet :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Max Marks** | **Obtained Marks** | **Comments(*if any*)** |
| 1. | 05 |  |  |
| 2. | 05 |  |  |
| 3. | 05 |  |  |
| 4. | 05 |  |  |
| **Total** | **20** |  | **Signature** |

**LOOPS**

There are three types of loops: while, until and for. The *while loop* is followed by a command or an expression enclosed in square brackets, a do keyword, a block of statements, and terminated with the done keyword. As long as the expression is true, the body of statements between do and done will be executed.

The *until loop* is just like the while loop, except the body of the loop will be executed as long as the expression is false.

The *for loop* used to iterate through a list of words, processing a word and then shifting it off, to process the next word. When all words have been shifted from the list, it ends. The for loop is followed by a variable name, the in keyword, and a list of words then a block of statements, and terminates with the done keyword.

The loop control commands are break and continue.

Syntax

While Loop

while command

do

block of statements  
 done

For Loop

for var in word1......word

do

Statement(s) to be executed for every word

done

Until Loop

Until command

do

Statement(s) to be executed until command is true

done

Example:

for var in 0 1 2 3 4 5 6 7 8 9

do

echo $var

done

**Task 1** Write a program to display the numbers from 10 to 20 in reverse order using for loop.

**Code**

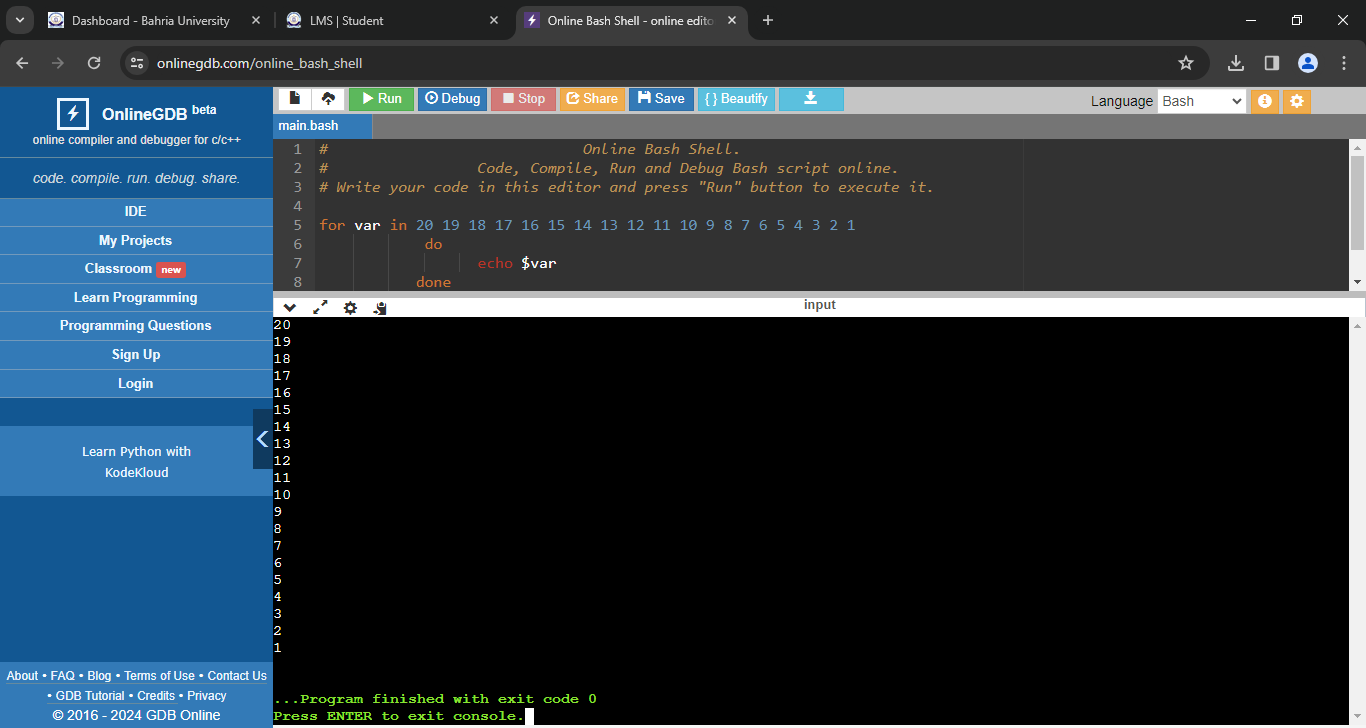
for var in 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

do

echo $var

done

**Output**



**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.

Example

#!/bin/bash

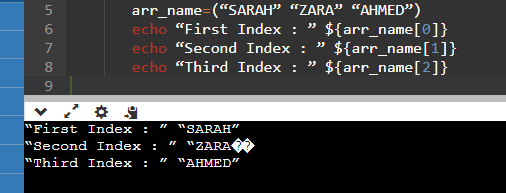
arr\_name=(“SARAH” “ZARA” “AHMED”)

echo “First Index : ” ${arr\_name[0]}

echo “Second Index : ” ${arr\_name[1]}

echo “Third Index : ” ${arr\_name[2]}

**OUTPUT**



**Task 2:** Write the output of the following Array programs. Array through positional parameters

#!/bin/bash

#Arrays through positional parameters

set apples bananas oranges peach

echo $1

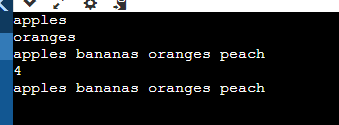
echo $3

echo $\*

echo $#

echo $@

**OUTPUT**



Declaring an array, adding elements one by one, displaying them one by one and displaying them all together as separate elements.

**Another way of using Arrays**

#!/bin/bash

Names=””

Names=”${Names} TheFirstNameHere”

Name=”${Names} TheSecondNameHere”

Name=”${Names} TheLastNameHere”

echo “Displaying one at a time ...”

for Names in ${Names};

do

echo ${Names};

done

echo “Only one element now”

echo ${Names}  
  
CODE:

Names=("Muhammad" "Hammad" "Mazhar")

Name=("${Names[0]}TheFirstNameHere" "${Names[1]}TheSecondNameHere" "${Names[2]}TheLastNameHere")

echo Displaying one at a time ...

for Namess in ${Name[@]};

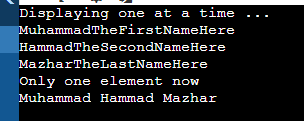
do

echo ${Namess};

done

echo Only one element now

echo ${Names[@]}**OUTPUT**



**FILE TESTING**

The Bourne shell uses the test command to evaluate conditional expressions and has a built-in set of options for testing attributes of files, such as whether it is a directory, a plain file (not a directory), a readable file, and so forth.

**Some options for testing attributes if file are:**

-d File is a directory  
-f File exists and is not a directory   
–r Current user can read the file  
–s File is of nonzero size  
–w Current user can write to the file   
–x Current user can execute the file

**Task 3** Write the output of the program for file testing.

#!/bin/bash

File=”Give the path of file here”

if [ -s $File ]

then

echo “File’s length is not zero”

else

echo “File’s length is zero”

fi

if [ -r $File ]

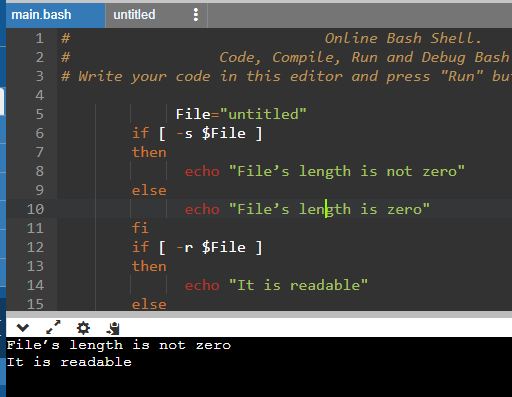
then

echo “It is readable”

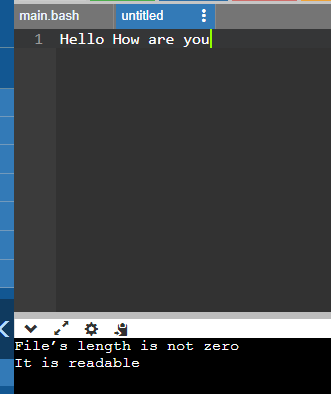
else

echo “It is not readable”

fi

**OUTPUT**

**FILE:**



**STRINGNS**

Strings are actually one-dimensional array of characters terminated by a null character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a null.

**Task 4**

Write the output for concatenation of two strings.

**Algorithm**

**Step 1** Enter into the vi editor and go to the insert mode for entering the code   
**Step 2** Read the first string.  
**Step 3** Read the second string  
**Step 4** Concatenate the two strings   
**Step 5** Enter into the escape mode for the execution of the result and verify the output.

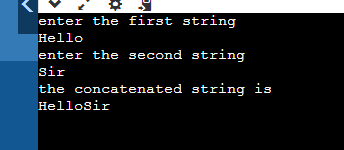
page54image12781184

**Program**

echo “enter the first string” read str1   
echo “enter the second string” read str2   
echo “the concatenated string is” $str1$str2

page54image12916832

**OUTPUT**



**Task 4.1** Write the program for the comparison of two strings.

**Algorithm**

**Step 1** Enter into the vi editor and go to the insert mode for entering the code   
**Step 2** Read the first string.  
**Step 3** Read the second string  
**Step 4** Compare the two strings using the if loop   
**Step 5** If the condition satisfies then print that two strings are equal else print two strings are not equal.   
**Step 6** Enter into the escape mode for the execution of the result and verify the output

**Program**

**Code:**

echo enter the first string

read str1

echo enter the second string

read str2

if [[ $str1 = $str2 ]]; then

echo "Both String 1 and String 2 is equal"

else

echo "String 1 and String 2 are not equal"

fi

**Output:**

