**RIPHAH INTERNATIONAL UNIVERSITY, ISLAMABAD**

****

**Lab 11**

**Bachelors of Computer science – 5th semester**

**Subject:** Operating System Lab

**Submitted to:** Ma’am Kausar Nasreen Khattak

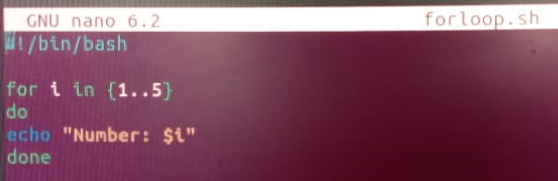
**Submitted by:** Munaza Malik

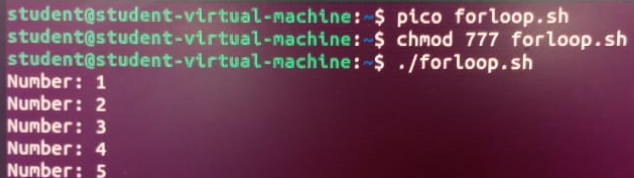
**Sap Id:** 44956

**Date:** 29th October, 2024

**Lab Tasks:**

**Q1.** How do you create a basic for loop in a shell script to iterate over a list of numbers from 1 to 5?



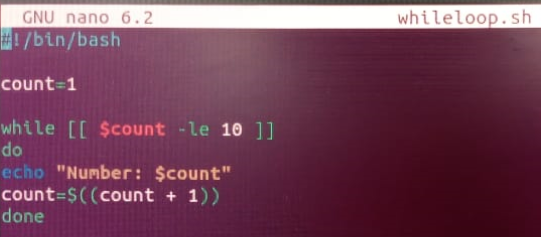


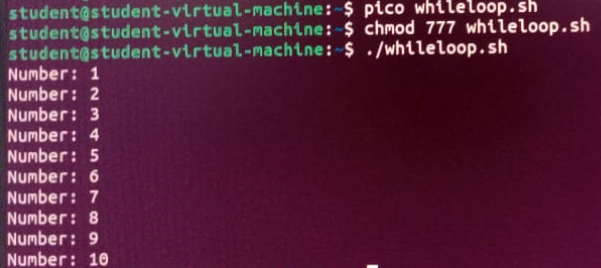
**for i in {1..5}**: This specifies the loop, where i takes each value in the range {1..5}. So i will sequentially be 1, 2, 3, 4, and 5.

**do ... done**: The do keyword starts the block of code to execute for each loop iteration, and done ends the loop.

**echo "Number: $i"**: This is the command within the loop, where $i is the current value in each iteration. This line prints the number currently assigned to i.

**Q2.** What is the syntax for a while loop that counts from 1 to 10 and prints each number?





**count=1**: Initializes the variable count to 1. This variable will keep track of the current count as the loop progresses.

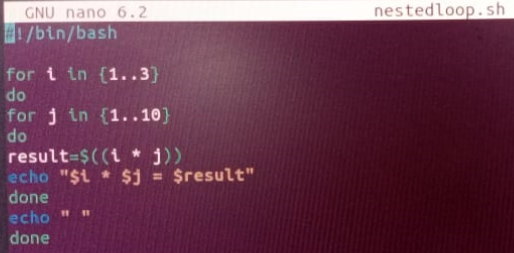
**while [ $count -le 10 ]**: The while loop checks the condition [ $count -le 10 ]. This means that as long as count is less than or equal to 10, the loop will continue executing.

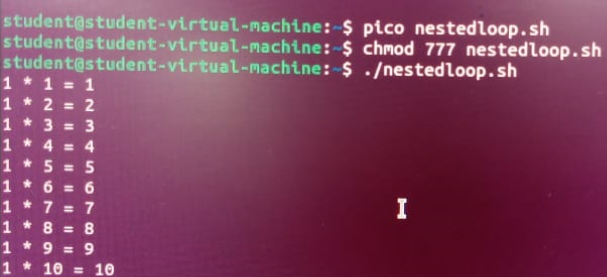
**do ... done**: The do keyword marks the beginning of the loop block, and done marks the end of it.

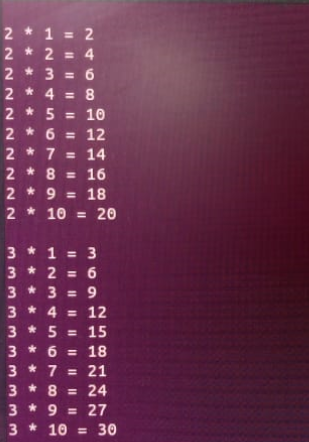
**echo "Number: $count"**: Inside the loop, this command prints the current value of count.

**count=$((count + 1)**: This increments the value of count by 1 after each iteration.

**Q3.** How would you write a nested loop in shell scripting to print a multiplication table from 1 to 3?







**for i in {1..3}**: This is the outer loop that iterates through each number from 1 to 3. The variable i represents the first number in the multiplication.

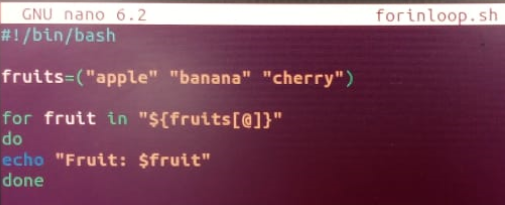
**for j in {1..10}**: This is the inner loop that also iterates through numbers 1 to 10. The variable j represents the second number in the multiplication.

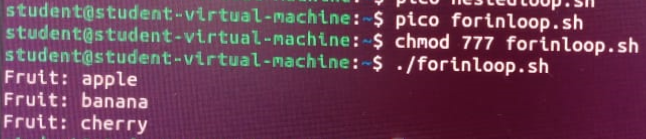
**result=$((i \* j))**: Inside the inner loop, we calculate the product of i and j using $(( ... )), which is the syntax for arithmetic operations in shell scripting.

**echo "$i \* $j = $result"**: This prints the multiplication expression and result.

**echo " "**: After the inner loop completes for a particular i, this line adds a space to distinguish each set of results for readability.

**Q4:** Write a for-in loop that iterates over the items in an array named fruits containing apple, banana, and cherry.





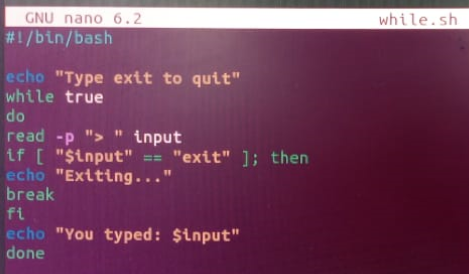
**fruits=("apple" "banana" "cherry")**: This defines an array named fruits with three elements: apple, banana, and cherry.

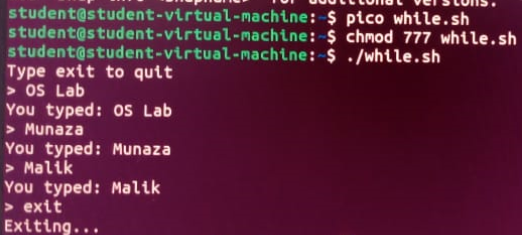
**for fruit in "${fruits[@]}"**: This starts the loop, where each item in the array fruits is assigned to the variable fruit for each iteration. The "${fruits[@]}" syntax expands the entire array, allowing for to loop through each element.

**do ... done**: The do keyword begins the block of code that will execute for each item, and done ends the loop.

**echo "Fruit: $fruit"**: This command prints the current item in the fruit variable, so each iteration displays one fruit from the array.

**Q5:** Write a while loop that reads user input until the user types "exit".





**while true**: This creates an infinite loop, which will continue to execute until explicitly broken with the break command. The keyword true makes the loop condition always true.

**read -p "> " input**: This line prompts the user for input and stores it in the variable input. The -p option allows you to specify a prompt message.

**if [ "$input" == "exit" ]; then ... fi**: This if statement checks if the value of input is "exit". If it is, it prints a message and uses break to exit the loop.

**echo "You entered: $input"**: If the user did not type "exit", this line will print the entered input.

**break**: When the user types "exit", this command ends the loop, causing the script to stop.