|  |  |
| --- | --- |
| Program- BTech-3rd Semester | Type- Sp. Core-I |
| Course Code- CSET213 | Course Name-Linux and Shell Programming |
| Year- 2024 | Semester- Odd |
| Date- 09/09/2022 | Batch- BXX-BXX (Cyber Security) |

**Lab Assignment 7**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Exp No** | **Name** | **CO1** | **CO2** | **CO3** |
| 7 | Shell programming loops | - | 100+ Free Tick Mark & Tick Images - Pixabay | - |

**Objective**: To learn and use shell loops for the development of applications.

**Outcomes:** After hands-on you will be able to write basic shell scripts using loops for the development of applications.

# Hands-on Learning on loops (60 minutes)

1. **For Loops: for loops iterate through a set of values until the list is exhausted**

**Example:**

**Method 1:**

#!/bin/bash

for i in 1 2 3 4 5

do

echo " Iteration number $i"

done

Latest bash version 3.0+ has inbuilt support for setting up ranges:

**Method 2:**

#!/bin/bash

for i in {1..5}

do

echo "Iteration $i times"

done

Bash v4.0+ has inbuilt support for setting up a step value using {START..END..INCREMENT} syntax:

**Method 3:**

#!/bin/bash

echo "Bash version ${BASH\_VERSION}..."

for i in {0..10..2}

do

echo "Iteration $i times"

done

**Method 4:**

#!/bin/bash

# set counter 'c' to 1 and condition

# c is less than or equal to 5

for (( c=1; c<=5; c++ ))

do

echo "Iteration $c times"

done

**Conditional exit with break**

for I in 1 2 3 4 5

do

statements1 #Executed for all values of ''I'', up to a disaster-condition if any.

statements2

if (disaster-condition)

then

break #Abandon the loop.

fi

statements3 #While good and, no disaster-condition.

done

**Early continuation with continue statement**

for I in 1 2 3 4 5

do

statements1 #Executed for all values of ''I'', up to a disaster-condition if any.

statements2

if (condition)

then

continue #Go to next iteration of I in the loop and skip statements3

fi

statements3

done

**Command substitution**

for var in $(command)

do

print "$var"

done

1. **while loop:** The bash while loop is a control flow statement that allows code or commands to be executed repeatedly based on a given condition.

while [ condition ]

do

command1

command2

command3

done

**Example:**

**Following while loop will print welcome 5 times on screen:**

#!/bin/bash

x=1

while [ $x -le 5 ]

do

echo "Welcome $x times"

x=$(( $x + 1 ))

done

1. **until loop:** The until loop is executed as many as times the condition/command evaluates to false. The loop terminates when the condition/command becomes true.

**Syntax:**

until [ expression ]

do

code block

...

...

Done

**Example:** Print and count the number starting with 1 and increment it by 1. When the count is equal to five, skip it. Similarly, the loop breaks when the count is equal to or greater than 10.

**Code:**

count=0

until false

do

((count++))

if [[ $count -eq 5 ]]

then

continue

elif [[ $count -ge 10 ]]

then

break

fi

echo "Counter = $count"

done

# Scripting Problems for Assessment (60 Minutes)

1. Write bash scripts to implement multiplication table of 36 using for loop. (15 Minutes)
2. Write bash scripts to calculate factorial of 16 using while loop. (15 Minutes)
3. Write bash scripts to append .out with each regular file in the present working directory. However, if .out is already existed with the file then skip it. (30 Minutes)

# Submission Instructions:

1. Submission requires the screen shots of all the incurred steps to execute a shell script.
2. Use the naming convention: Prog\_CourseCode\_RollNo\_LabNo.docx
3. Submission is through LMS only