School of Computer Science Engineering and Technology
Bennett University (The Times Group)

Program- BTech-3rd Semester
Course Code- CSET213
Year- 2025
Date- 29/08/2025

Type- Sp. Core-I
Course Name-Linux and Shell Programming
Semester- Odd
Batch- Cyber Security (B75-86)

Lab Assignment 6

Exp No	Name	CLO Achieved				Marks
		CO1	CO2	CO3	CO4	
6	Shell Operators	√	√			2

Objective: To implement the operators using bash scripts

Outcomes: After hands-on you will be able to write basic shell scripts implement arithmetic, relational, logical, and bitwise operators.

Hands-on Learning on operators (60 minutes)

Like other programming languages, shell programming also supports various types of operators to perform tasks. Operators can be categorized as follows:

- Assignment operator (=)
- Arithmetic operators
  - +,
  - –
  - *
  - /
  - %
  - ** (exponentiation)
  - += (plus-equal)
  - -= (minus-equal)
  - *= (multiplication-equal)
  - /= (slash-equal)
  - %= (mod-equal)
- Logical operators
  - **!** (**NOT**)
  - **&&** (**AND**)
  - || (**OR**)
- Comparison operators
  - -eq (is equal to) - [$a -eq $b]
  - -ne (is not equal to) - [$a -ne $b]
  - -gt (is greater than) - [$a -gt $b]
  - -ge or >= (is greater than or equal to) - [$a -ge $b]
  - -lt (is less than) - [$a -lt $b]
  - -le (is less than or equal to) - [$a -le $b]
  - < (is less than) - (($a < $b))
  - <= (is less than or equal to) - (($a <= $b))
  - > (is greater than) - (($a > $b))
  - >= (is greater than or equal to) - (($a >= $b))

- **Redirection Operators**
  - **Output redirection operator (>):** $ `command > outputfile, Example: $ date > test6`
  - **Input redirection operator (<):** $ `command < inputfile, $ wc < test6`
  - **Inline input redirection (>>):** only on command line, not in a file $ `command << marker`

  Example:
```
$ wc << EOF
> test string 1
> test string 2
> test string 3
> EOF
```

- **Mathematical operators:** Number manipulation in shell scripts is done using two methods:
  a. The expr command (for compatibility with Bourne shell)
  b. Using brackets (Better way)

The expr command: `$ expr 1 + 5`

TABLE 11-1 The expr Command Operators

Operator	Description
ARG1 | ARG2	Returns ARG1 if neither argument is null or zero; otherwise, returns ARG2
ARG1 & ARG2	Returns ARG1 if neither argument is null or zero; otherwise, returns 0
ARG1 < ARG2	Returns 1 if ARG1 is less than ARG2; otherwise, returns 0
ARG1 <= ARG2	Returns 1 if ARG1 is less than or equal to ARG2; otherwise, returns 0
ARG1 = ARG2	Returns 1 if ARG1 is equal to ARG2; otherwise, returns 0
ARG1 != ARG2	Returns 1 if ARG1 is not equal to ARG2; otherwise, returns 0
ARG1 >= ARG2	Returns 1 if ARG1 is greater than or equal to ARG2; otherwise, returns 0
ARG1 > ARG2	Returns 1 if ARG1 is greater than ARG2; otherwise, returns 0
ARG1 + ARG2	Returns the arithmetic sum of ARG1 and ARG2
ARG1 - ARG2	Returns the arithmetic difference of ARG1 and ARG2
ARG1 * ARG2	Returns the arithmetic product of ARG1 and ARG2
ARG1 / ARG2	Returns the arithmetic quotient of ARG1 divided by ARG2
ARG1 % ARG2	Returns the arithmetic remainder of ARG1 divided by ARG2
STRING : REGEXP	Returns the pattern match if REGEXP matches a pattern in STRING
match STRING REGEXP	Returns the pattern match if REGEXP matches a pattern in STRING
substr STRING POS LENGTH	Returns the substring LENGTH characters in length, starting at position POS (starting at 1)
index STRING CHARS	Returns position in STRING where CHARS is found; otherwise, returns 0
length STRING	Returns the numeric length of the string STRING
+ TOKEN	Interprets TOKEN as a string, even if it's a keyword
(EXPRESSION)	Returns the value of EXPRESSION

Example: Write a script to print the quotient of expression var1/var2 using *expr command*.

```
#!/bin/bash
# An example of using the expr command
var1=10
var2=20
var3=$(expr $var2 / $var1)
echo The result is $var3
```

Using brackets: (`$[ operation ]`):

```
$ var1=$[1 + 5]
$ echo $var1
6
$ var2=$[$var1 * 2]
$ echo $var2
12
```

Limitation: The bash shell supports only integer arithmetic. We can bash calculator (**bc)** floating point arithmetic.

Example: Write a script to implement floating point arithmetic using **bc**

```
#!/bin/bash
var1=$(echo "scale=4; 3.44 / 5" | bc)
echo The answer is $var1
```

Scripting Problems for Assessment (60 Minutes)

1. Write a script that:
    a. Assigns two numbers to variables a=15 and b=4
    b. Prints the results of addition, subtraction, multiplication, division, modulus, and exponentiation.
2. Write a script that:
    a. Initializes num=10
    b. Uses +=, -=, *=, /=, %= operators step by step to modify num
    c. Prints the value of num after each operation.
3. Write a script that:
    a. Sets x=25 and y=50
    b. Prints whether x is equal, not equal, greater, less, greater-or-equal, and less-or-equal compared to y using relational operators (-eq, -ne, -gt, -lt, -ge, -le).
    (Hint: Just print the evaluated result, e.g., echo "$x -eq $y → [$x -eq $y]")
4. Write a script that:
    a. Defines two variables a=5, b=10
    b. Prints the result of the following logical operations:
        i. [$a -lt $b] && echo "true"
        ii. [$a -gt $b] || echo "true"
        iii. ! [$a -eq $b] && echo "true"
5. Write a script that:
    a. Uses date > today.txt to redirect current date into a file.
    b. Uses wc < today.txt to count lines, words, and characters.
    c. Uses echo "Extra Line" >> today.txt to append a line.
    d. Finally, displays the contents of today.txt.
6. Create a single, robust Bash script named bitwise_calc.sh that functions as a command-line calculator for bitwise operations. The script must parse user input from command-line arguments, perform the specified calculation, and display the result in a clear, human-readable format.
    a. The script must accept **three command-line arguments** for binary operations (AND, OR, XOR, Left Shift, Right Shift): `number1`, `operator`, `number2`.
    b. For the unary operation (Complement), the script must accept **two arguments**: `operator`, `number1`.
    c. All numerical inputs will be non-negative integers.

Example of script operation:

./bitwise_calc.sh 10 & 12
./bitwise_calc.sh 25 | 10
./bitwise_calc.sh 45 ^ 33
./bitwise_calc.sh 8 '<<' 2
./bitwise_calc.sh 16 '>>' 3
./bitwise_calc.sh ~ 5

Submission Instructions:

1. Submission requires the screen shots of all the incurred steps to execute a shell script or a video showing the whole process.
2. All these files are in single zip folder.
3. Use the naming convention: Prog_CourseCode_RollNo_LabNo.docx (Example: BCA3rdSem_CBCA221_ E21BCA002_Lab1.1)
4. Submission is through LMS only