SHELL

What’s Shell?

It acts an interface between the user and OS (kernel).

It’s known as “ command interpreter”.

When you type ls :

shell finds cmd (/usr/bin).

shell runs cmd.

you receive the output.

What’s Shell Program?

- It’s collections of executables or commands placed in a file and executed.
- It provides user an option to execute a command based on some condition.
- It provides conditional and control statements (if,for,while,switch-case etc)

Shell Types

In UNIX there are two major types of shells:

1. The Bourne shell. If you are using a Bourne-type shell, the default prompt is the $ character.
2. The C shell. If you are using a C-type shell, the default prompt is the % character.

There are again various subcategories for Bourne Shell which are listed as follows:

- Bourne shell (sh)
- Korn shell (ksh)
- Bourne Again shell (bash)

The different C-type shells follow:

- C shell (csh)
- TENEX/TOPS C shell (tcsh)

Shell Scripts

- The basic concept of a shell script is a list of commands, which are listed in the order of execution.
- This would be a simple text file in which we would put our all the commands and several other required constructs that tell the shell environment what to do and when to do it.

```
#print date and time -  today.sh
echo "Today is:"
date
```

Save it as today.sh

Run:

```
sh today.sh
```

```
echo "What is your name?"
read PERSON
echo "Hello, $PERSON"
```

Here is sample run of the script:

```
$ sh filename.sh

What is your name?
Alex
Hello, Alex
```

VARIABLES

Variables

- A variable is a character string to which we assign a value. The value assigned could be a number, text, filename, device, or any other type of data.

Defining Variables

- Variables are defined as follows:

SYNTAX:

```
variable_name=variable_value
```

Example:

```
NAME=“KIIT"
VAR1=100
```

Variables of this type are called scalar variables.

A scalar variable can hold only one value at a time.

Accessing Values

- To access the value stored in a variable, prefix its name with the dollar sign ($):
- Example:

```
NAME=“KIIT"
echo $NAME
```

Output:

```
KIIT
```

Read-only Variables

- The shell provides a way to mark variables as read-only by using the readonly command. After a variable is marked read-only, its value cannot be changed.
- Example:

```
NAME=“KIIT"
readonly NAME
NAME=“University"
```

This would produce following result:

```
/bin/sh: NAME: This variable is read only.
```

Unsetting Variables

- Unsetting or deleting a variable tells the shell to remove the variable from the list of variables that it tracks. Once you unset a variable, you would not be able to access stored value in the variable.
- Syntax:

```
unset variable_name
```

Example:

```
NAME="Zara Ali"
unset NAME
echo $NAME
```

Above example would not print anything.

ARRAY

- Arrays provide a method of grouping a set of variables. Instead of creating a new name for each variable that is required, you can use a single array variable that stores all the other variables.

Defining Array Values

- Syntax: array_name[index]=value
- Example:

```
NAME[0]=“KIIT"
NAME[1]=“University"
```

Accessing Array Values

- Syntax: ${array_name[index]}
- Example:

```
NAME[0]=“KIIT"
NAME[1]=“University“
echo "First Index: ${NAME[0]}"
echo "Second Index: ${NAME[1]}"
```

- $./test.sh

```
First Index: KIIT
Second Index: University
```

- You can access all the items in an array in one of the following ways:
- Syntax:

```
${array_name[*]}
${array_name[@]}
```

- Example:

```
NAME[0]=“KIIT"
NAME[1]=“University“
echo "First Method: ${NAME[*]}"
echo "Second Method: ${NAME[@]}“
```

- $./test.sh

```
First Method: KIIT University
Second Method: KIIT University
```

Basic Operators

There are following operators

- Arithmetic Operators.
- Relational Operators.
- Boolean Operators.
- String Operators.

- Example:

```
val=`expr 2 + 2`
echo "Total value : $val"
```

Output:

```
Total value : 4
```

Arithmetic Operators

```
a=10
b=20
val=`expr $a + $b`
echo "a + b : $val"
val=`expr $a - $b`
echo "a - b : $val"
val=`expr $a \* $b`
echo "a * b : $val“
val=`expr $b / $a`
echo "b / a : $val"
val=`expr $b % $a`
echo "b % a : $val"
```

```
if [ $a -eq $b ]
then
echo "a is equal to b"
fi
if [ $a -ne $b ]
then
echo "a is not equal to b"
fi
```

- Output:

```
a + b : 30
a - b : -10
a * b : 200
b / a : 2
b % a : 0
a is not equal to b
```

Relational Operators

```
a=10
b=20
if [ $a -eq $b ]
then
echo "$a -eq $b : a is equal to b"
else
echo "$a -eq $b: a is not equal to b"
fi
if [ $a -ne $b ]
then
echo "$a -ne $b: a is not equal to b"
else
echo "$a -ne $b : a is equal to b"
fi
```

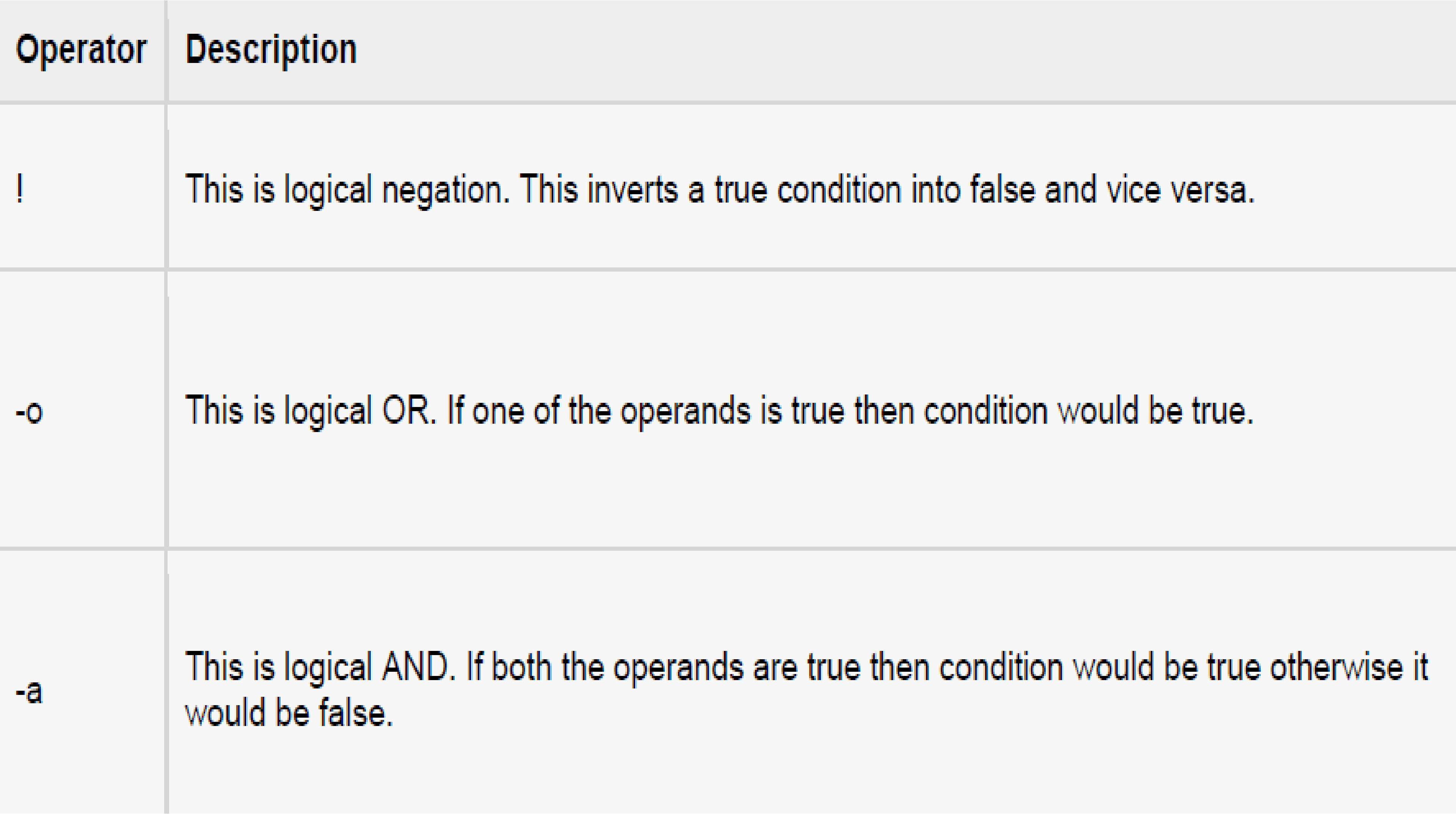
```
if [ $a -gt $b ]
then
echo "$a -gt $b: a is greater than b"
else
echo "$a -gt $b: a is not greater than b"
fi
if [ $a -lt $b ]
then
echo "$a -lt $b: a is less than b"
else
echo "$a -lt $b: a is not less than b"
fi
```

```
if [ $a -ge $b ]
then
echo "$a -ge $b: a is greater or equal to b"
else
echo "$a -ge $b: a is not greater or equal to b"
fi
if [ $a -le $b ]
then
echo "$a -le $b: a is less or equal to b"
else
echo "$a -le $b: a is not less or equal to b"
fi
```

- Output:

```
10 -eq 20: a is not equal to b
10 -ne 20: a is not equal to b
10 -gt 20: a is not greater than b
10 -lt 20: a is less than b
10 -ge 20: a is not greater or equal to b
10 -le 20: a is less or equal to b
```

Boolean Operators



Operator	Description
!	This is logical negation. This inverts a true condition into false and vice versa.
-o	This is logical OR. If one of the operands is true then condition would be true.
-a	This is logical AND. If both the operands are true then condition would be true otherwise it would be false.

```
a=10
b=20

if [ $a != $b ]
then
echo "$a != $b : a is not equal to b"
else
echo "$a != $b: a is equal to b"
fi

if [ $a -lt 100 -a $b -gt 15 ]
then
echo "$a -lt 100 -a $b -gt 15 : returns true"
else
echo "$a -lt 100 -a $b -gt 15 : returns false"
```

```
if [ $a -lt 100 -o $b -gt 100 ]
then
echo "$a -lt 100 -o $b -gt 100 : returns true"
else
echo "$a -lt 100 -o $b -gt 100 : returns false"
fi

if [ $a -lt 5 -o $b -gt 100 ]
then
echo "$a -lt 100 -o $b -gt 100 : returns true"
else
echo "$a -lt 100 -o $b -gt 100 : returns false"
fi
```

- Output:

```
10 != 20 : a is not equal to b
10 -lt 100 -a 20 -gt 15 : returns true
10 -lt 100 -o 20 -gt 100 : returns true
10 -lt 5 -o 20 -gt 100 : returns false
```

Case-Esac Statement

```
case word in
  pattern1)
    Statement(s) to be executed if pattern1 matches
    ;;
  pattern2)
    Statement(s) to be executed if pattern2 matches
    ;;
  pattern3)
    Statement(s) to be executed if pattern3 matches
    ;;
  *)
   Default condition to be executed
   ;;
esac
```

Case-Esac Statement

```
FRUIT="kiwi"

case "$FRUIT" in
   "apple") echo "Apple pie is quite
tasty."
   ;;
   "banana") echo "I like banana nut
bread."
   ;;
   "kiwi") echo "New Zealand is famous
for kiwi."
   ;;
esac
```

Case-Esac Statement

```
echo “Enter a number”
read num
case $num in
[0-9]) echo “you have entered a single digit number”
 ;;
[1-9][1-9]) echo “you have entered a two-digit number”
 ;;
[1-9][1-9][1-9]) echo “you have entered a three-digit number”
 ;;
*) echo “your entry does not match any of the conditions”
 ;;
Esac
```

while loop – syntax

```
while [ condition ]
do
    code block;
done
```

```
#while_ex.sh
verify="n"
while [ "$verify" != y ]
do
    echo "Enter option: "
    read option
    echo "You entered $option.  Is this correct? (y/n)"
    read verify
done
```

```
#simple for loop
for i in 1 2 3
do
echo "==>$i"
Done
```

```
#simple for loop
for (( j = 1 ; j <= 5; j++ ))
    do
         echo "$j "
    done
```

Assignment:

1. WAP to swap the values of two numbers.
2. WAP to perform addition, subtraction, multiplication, division and modulus of two numbers.
3. WAP to check whether a number is even or odd.
4. WAP to print the largest number among three numbers.
5. WAP to implement grading system.
6. Write a shell program to find whether a given year is a leap year or not.

Lab Experiments

1. WAP to print numbers between 1 to 10.
2. Write a shell script to display the gross salary of an employee (basic+da+hra).
3. Write a shell script to which will accept a number & find out the summation of square of last 3 digits.
4. Write a shell script to find out the electrical bill amount for consumer according to different unit charges.
5. Write a shell script to display 10 numbers it using an array.
6. Write a shell script to find out maximum and minimum element from given array of elements.
7. Write a shell script to display location of an element in an array.

Lab Experiments

8. Write a shell script to merge content of two different arrays.
9. Write a shell script to sort an array of 10 numbers.
10. Write a shell script to insert & delete from a particular location in an given array of elements.
11. Write a shell script to delete duplicate elements from a given array of elements.
12. Write a shell script to display elements of an array in reverse order.
13. Write a shell script to display the 1st & 2nd element from a given array of elements.
14. Write a shell script to calculate the overtime (Hours) payment of an employee as per rules.
15. Write a shell program to evaluate the operation $1^2+2^2+3^2+......+n^2$

Program Implementation Activities

1. Write a shell script to display the alternate digits in a given seven
   digits number starting first digit.
2. Write a shell script to print all the even odd between 0 to 100
3. Write a shell script to print factorial of a given number.
4.Write a shell script to print Fibonacci series starting from 0.
5.Write a shell script to print a number in reverse order & calculate
its sum of its digits.
6. Write a shell script to find (check whether) palindrome numbers
in a given range.
7. Write a shell script to print the prime numbers in a given range.
8. Write a shell script to find (check whether) Armstrong numbers
in a given range.
9. Write a shell script to convert decimal number to binary number.
10. WAP to implement grading system.