SHELL PROGRAMMING

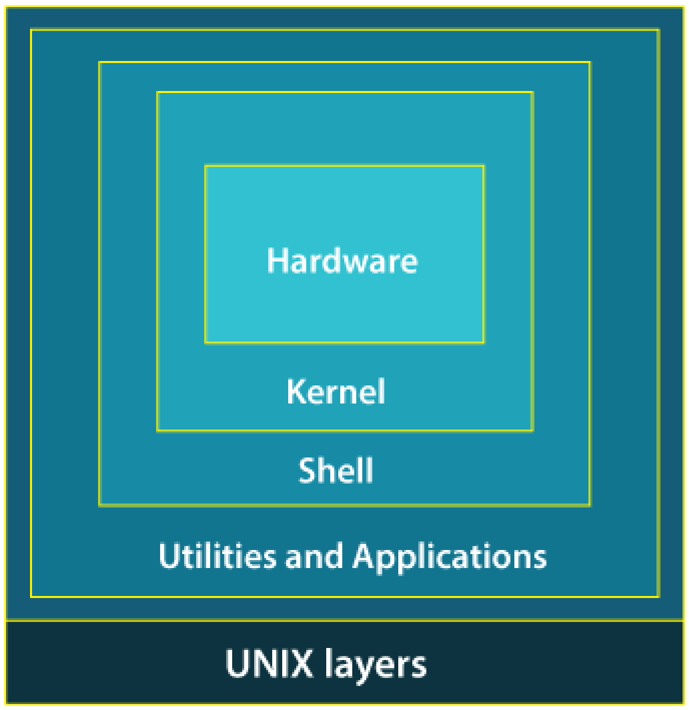
Objective:

To study the concept of shell program in UNIX .

Concepts Involved:

Shell programming is a group of commands grouped together under single filename. The shell interprets the input, takes appropriate action, and finally displays the output. Shell scripts are dynamically interpreted, not compiled.

UNIX Architecture:



Types of Shell:

Bourne shell	sh
C shell	csh
Korne Shell	ksh

Creation and execution of shell scripts using command line editor:

1. creation
```
$ cat > greet
echo "please enter your name:"
read name
echo "hi! Welcome to this session $name"
Ctrl + D
```
2. Execution

```
$ sh greet
please enter your name: java
hi! Welcome to this session java”
```

(OR)

vi Editor

Valid shell variables:

n

area
a1
account
a_count

Assigning values to variable:

Variable=value

Displaying values of variables:

```
$ echo value of n is $n
```

Operators:

Arithmetic Operators provided by the shell are +,- * and /

1. Logical operators

-a	and
-o	or
!	not

2. Relational operators

-eq	: check fro equality of integers
-ne	: check for inequality
-gt	: check if one integer is greater than the other
-lt	: check if one integer is lesser than the other
-ge	: check if one integer is greater than or equal to the other
-le	: check if one integer is lesser than or equal to the other.
-f	: check if a file is an ordinary file
-d	: check if a file is a directory
-r	: check if a file is readable
-w	: check if a file is write able
-x	: check if a file is executable

3. String comparison operators

=	equal to
!=	not equal to

4. Conditional execution operations

&& used to execute a command on successful execution of another command.

|| used to execute another command on failure of another command.

5. Read command

Used to read the value of the shell variable from a user.

syntax:

read name

6. Comment statement

# this is a text program.

7. Programming Language Control Construct

1.a)if..then…else…fi   b) if..then..elif..else ..fi

2.for…do…done

3.while..do..done

4.until…do..done

5.case …esac

1) <u>if construct</u>

if construct is useful for executing a set of commands based on the condition being true and alternate set of commands to be executed if the condition is false.

```
 Ex. if (grep India countri.dat)

     then
          echo "pattern found"
     else
          echo "pattern not found"
     fi
```

<u>2) for construct</u>

It is used to perform same set of operations on a list of values.

```
for variable in value1 value2 value3 …
do

Commands
done

Ex. for k in 1 2 3 4 5
     do
          echo "the number is $k"
```

```
        echo “the square of the number is `expr $k \*
    $k` “ done
```

3)while construct

Repeatedly executing group of commands as long as the condition is true.

```
while condition
do

Commandlist

done
```

Ex.To print 3 numbers

```
a=1
while [$a -le 3]
do
echo $a
$a=`expr $a+1`
done
o/p. 1 2 3
```

4) until construct

Repeatedly executing group of commands until a condition is true.

Syntax:

```
until condition
do
Commandlist
done
```

Ex. To print 3 numbers

```
a=1
until [$a -le 3]
do
echo $a
$a=`expr $a+1`

done

o/p. 1 2 3
```

5) case construct:

Syntax :

```
case value in
choice1) commands;;
choice2)commands;;
….
esac
Ex.     $echo “enter a value”
            read myval
```

```
case “$myval” in

0)   echo zero;;
1)   echo one;;
2)   echo two;;
3)   echo three;;
*) echo “invalid argument”;;
esac
```