4th Part – Control Statements

If Syntax

```
if [ expression ];
then
    statement
fi
```

main.sh

```
#!/bin/bash
read -p "Enter a number: " abc
if [ $abc -gt 10 ];
then
    echo "The number is greater than 10."
fi
```

O/p ./main.sh

Enter a number : 11

The number is greater than 10

O/p

Enter a number : 9

no output

If else syntax

```
if [expression];

then
   statement 1

else
   statement 2

fi
```

main.sh

```
#!/bin/bash

read -p "Enter no : " number

if [ $number -gt 10 ];

then
   echo " great 10 "

else
   echo " less 10 "

fi
```

O/p ./main.sh

Enter no : 90

Great 10

O/p ./main.sh

Enter no : 8

less 10

what if we exactly 10.. it consider else statement as 10 is not greater or less. so it runs the else block and it won't show error

To counter this (elif usage)

$number -gt 10

```
if [ expression1 ];
then
    statements
elif [ expression2 ];
then
    statements
else
    statements
fi
```

$number -eq 10

now it check for input 10. we will see the expected results.

Nested if (Syntax)

```
if [ expression1 ];
then
    statements
else
    if [ expression 2 ];
    then
        statements
    fi
fi
```

For every if statement there will be a closing fi statement in linux and also have a then statement below it.

elif will have then statement but not fi statement.

Switch cases:

Syntax

```
case I/p
    Pattern 1) statement 1;;
    Pattern n) statement n;;
esac
```

main.sh

```
#!/bin/bash
echo "Enter no bet 1 to3:"
read n
case $n in
    1) echo "One";;
    2) echo "Two";;
    3) echo "Three";;
    *) echo "Invalid-other";;
esac

echo "vechile"
read V
case $V in
    "car") echo "BMW"
           echo "Spacious";;
```

→ denotes end of statement for the case

"bike") echo "Small";;

esac

loop

- while
- for
- until

while

```
while <condition>

do
    <command 1>
    <command 2>
    <etc>

done
```

main.sh

```
#!/bin/bash

a=0
while [ $a -lt 10 ];

do
    echo $a
    a=`expr $a + 1`

done
```

O/P ./main.sh

```
0
1
2
3
4
5
6
7
8
9
```

while loop stops when condition false.

Until loop

```
a=1
until [ $a -gt 5 ];
do
    echo $a
    a='expr $a + 1'
done
```

o/p

1
2
3
4
5

Until loop stops when condition is true.

When condition in while loop doesn't fail then it is called Infinite loop.

For loop

```
for <var> in < value1 value2 ... valuen>
do
   <command 1>
   <command 2>
   <etc>
done
```

main.sh

```
#! /bin/bash
for a in 1 2 3 4 5 6 7 8 9 10;
do
   echo "Iteration no $a"
done
```

To handle large Range

```
for a in {1..50};
do
   echo "Iteration no $a"
done
```

break / continue

main.sh

```
#! bin/bash
for a in 1 2 3 4 5 6 7 8 9 10;
do
   if [ $a -eq 5 ];
   then
      break (or) continue
   else
      echo $a
   fi
done
```

break

o/p ./main.sh

1
2
3
4

continue

o/p ./main.sh

1
2
3
4
6
7
8
9
10

while, for, until block will always be accompanied with do and done statements.

— x — x — x —