TATA CONSULTANCY SERVICES

Experience certainty.

Confidentiality Statement

Tata Code of Conduct

We, in our dealings, are self-regulated by a Code of Conduct as enshrined in the Tata Code of Conduct. We request your support in helping us adhere to the Code in letter and spirit. We request that any violation or potential violation of the Code by any person be promptly brought to the notice of the Local Ethics Counselor or the Principal Ethics Counselor or the CEO of TCS. All communication received in this regard will be treated and kept as confidential.

TATA CONSULTANCY SERVICES

TATA CONSULTANCY SERVICES
Experience certainty.

Virtual ILP – Shell Programming

Content Manual Version 1.1

December 2014

(ILP Guwahati)

Table of Content

1. Shell Decision Statement

While writing a shell script, there may be a situation when you need to adopt one path out of the given two paths. So you need to make use of conditional statements that allow your program to make correct decisions and perform right actions.

Unix Shell supports conditional statements which are used to perform different actions based on different conditions. Here we will explain following two decision making statements:

- The if...else statements
- The case...esac statement

1.1. If..else statements

We can use "if..else" statement which can be used as decision making statement to select an option from a given set of options.

Unix Shell supports following forms of if..else statement:

- if...fi statement
- if...else...fi statement
- if...elif...else...fi statement

Syntax:

- **if...fi statement**

```
if [ condition(s) ]
    then
            command(s)
    fi
```

- **if...else...fi statement**

```
if [ condition(s) ]
    then
    command(s)
else                    # Optional else part
    command(s)
```

```
fi
```

- **if...elif...else...fi statement**

```
if [ condition(s) ]
        then
        command(s)
elif [ condition(s) ]   # Many elif-then allowed
        then
        command(s)
else
        command(s)
fi
```

Example:

```
a=2;

if [ $a -ne 2 ]
then
        echo "value of a is not 2"
else
        echo "value of a is 2"
fi
```

Output:

Value of a is 2

1.2. test command in if condition

We can use test command as condition of if condition as mentioned below.

```
# cat if_test.sh

echo "want to quit (type Y or y):"
read ans

if test $ans = 'y' -o $ans = 'Y'
    then
     exit
else
        echo "Entered not n or N"
fi
```

1.3. case...esac Statement

We can use multiple if...elif statements to perform a multiway branch. However, this is not always the best solution, especially when all of the branches depend on the value of a single variable.

Unix Shell supports case...esac statement which handles exactly this situation, and it does so more efficiently than repeated if...elif statements.

The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used.

Syntax:

```
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
    ;;
esac
```

Here string word is compared against every pattern until a match is found. The statement(s) following the match pattern executes. If no matches are found, the case statement exits without performing any action. When statement(s) part executes, the command **;;** indicates program flow should jump to the end of the entire case statement.

There is no maximum number of patterns, but the minimum is one.

Example:

```
#!/bin/sh
COURSE=”DB”
Case “$COURSE” in
      “Java”) echo  “Java is a programming language”
      ;;
      “Perl”)echo “Perl is scripting language”
      ;;
      “DB”)echo “Oracle is a DB”
      ;;
esac
```

Output:
Oracle is a DB

2. Iterative Statements/Loop

Loops are a powerful programming tool that enables you to execute a set of commands repeatedly.

- While loop
- for loop
- until loop

2.1. while loop

Here condition is evaluated by shell. If the resulting value is true, given command(s) are executed. If condition is false then no command would be executed and program would jump to the next line after done statement.

Syntax:

```
while condition
do          # executed as long as the condition is true
   command(s)
done
```

Example:

```
a=0
while [ $a -lt 3 ]
do
     echo $a
     a=`expr $a + 1`
done
```

Output:

```
0
1
2
```

2.2. until loop

Here condition is evaluated by shell. If the resulting value is false, given command(s) are executed. If condition is true then no command would be executed and program would jump to the next line after done statement.

Syntax:

```
until   condition              # complement of while
do              # executed as long as the condition is false

  command(s)

done
```

Example:

```
a=0
until [ ! $a -lt 3 ]
do
        echo $a
        a=`expr $a + 1`
done
```

Output:

```
0
1
2
```

2.3. for loop

For loop operate on lists of items. It repeats a set of commands for every item in a list.

Syntax:

```
for var in word1 word2 ... wordN
do
        Statement(s) to be executed for every word.
done
```

Example:

```
for var in 0 1 2 3 4 5
do
    echo $var
done
```

Output:

```
0
1
2
3
4
5
```

Example:

```
for filename in `ls tempdir`
do
        echo “Displaying contents of ${filename}”
        cat ${filename}
done
```

We can use "break" and "continue" commands to control the loop. Command "break" causes the termination of a loop and continue resumes execution at its top.