LECTURE 3: COMMAND SUBSTITUTION AND CONDITIONAL EXECUTION

3.0 SUBSTITUTION

The shell performs substitution when it encounters an expression that contains or more special characters.

For example:

In a situation where the printing value of the variable is substituted by its value. Same time, "\n" is substituted by a new line:-

```
#!/bin/bash
z = 30
echo  –e "The value of z is  $z \n"
```

Here the –e option enables the interpretation of the backslash escapes.

The value of z is 30

It should be noted that the without the –e option, the output will be:-

The value of z is 30\n

3.1 ESCAPE AND DESCRIPTION

The following escape sequences which can be used in echo command:-

S/N	Escape and description
1	\\ backlash
2	\a Alert (BEL)
3	\b backspace
4	\c Suppress trailing newline
5	\f Form feed
6	\n

	New line
7	\r Carriage return
8	\t Horizontal tab
9	\v Vertical tab

You can use –E option to disable the interpretation of the backslash escape (default).

-n option can be used to disable the insertion of a new line.

3.2 UNIX/LINUX: CONDITIONAL STATEMENT

There are situation when one may need to adopt one out of the given two or more paths, while writing a shell script. In this situation, the use of conditional statements is recommended, so that the program can make correct decisions and perform the right actions.

Unix shell supports conditional statements which are used to perform different actions based on different conditions. The two decision-making statements are:-

If…else statement
Case … esac statement

3.2.1 The if … else Statements

If else statements are useful decision-making statements which can be used 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

Example 1

```
#!/bin/bash
echo –n "Enter one number"
read character
```

```
if [ “$character” = “1”];
then
echo “You entered one.”
else
echo “You did not enter one”
fi
```

Example 2

```
#!/bin/bash
n=10
if [ $n -lt 10 ];
then
echo "It is a one digit number"
else
echo "It is a two digit number"
fi
```

Example 3

```
#!/bin/bash
echo –n “Enter a number between 4 and 7 ”
read character
if [ “$character” = “4”]; then
        echo “You entered four.”
elif [ “$character” = “5”]; then
        echo “You entered five.”
elif [ “$character” = “6”]; then
        echo “You entered six.”
elif [ “$character” = “7”]; then
        echo “You entered seven.”
else
        echo “You did not enter a number between 4 and 7.”
fi
```

3.2.2 The case … esac Statement

Multiple **if … elif** statements can be used to perform a multiway branch. However, this won’t always be the best solution, especially when all of the branches depend on the value of a single variable. The Unix Shell supports **case … esac** statement which handles exactly this situation. The **case … esac** does it more efficiently than repeated **if … elif** statements.

Note: The **case … esac** statement in the Unix shell is very similar to the switch … case statement used in other programming languages like C or C++ and PERL, etc.

Example 4

```
#!/bin/bash
echo –n "Enter a number between 4 and 7 "
read character
case $character in
4)
echo "You entered four." ;;
5)
echo "You entered five." ;;
6)
echo "You entered six." ;;
7)
echo "You entered seven." ;;
* )
echo "You did not enter a number between 4 and 7."
esac
```

case also selectively executes statements if word matches a pattern. Any number of patterns and statements can be accommodated. The patterns can be literal text or wildcards. Multiple patterns separated by the "|" character are used.

Example 5

```
#!/bin/bash
echo –n "Enter a letter or digit "
read character
case $character in
                    # Check for letters
[ [ : lower: ] ] | [ [ : upper : ] ] ) echo " You entered letter $character"
                                        ;;
                    # Check for digits
[ 0 – 9 ] )                 echo " You entered digit $character"
                                        ;;
* )                                  echo "You did not enter a letter or a digit."
esac
```