UNIX / LINUX - SHELL SUBSTITUTION

What is Substitution?

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

Example

Here, the printing value of the variable is substituted by its value. Same time, **"\n"** is substituted by a new line –

Live Demo

```
#!/bin/sh

a=10
echo -e "Value of a is $a \n"
```

You will receive the following result. Here the **-e** option enables the interpretation of backslash escapes.

```
Value of a is 10
```

Following is the result without **-e** option –

```
Value of a is 10\n
```

The following escape sequences which can be used in echo command –

Sr.No.	Escape & Description
1	**** backslash
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 the **-E** option to disable the interpretation of the backslash escapes (default).

You can use the **-n** option to disable the insertion of a new line.

Command Substitution

Command substitution is the mechanism by which the shell performs a given set of commands and then substitutes their output in the place of the commands.

Syntax

The command substitution is performed when a command is given as –

```
`command`
```

When performing the command substitution make sure that you use the backquote, not the single quote character.

Example

Command substitution is generally used to assign the output of a command to a variable. Each of the following examples demonstrates the command substitution –

Live Demo

```
#!/bin/sh

DATE=`date`
echo "Date is $DATE"

USERS=`who | wc -l`
echo "Logged in user are $USERS"

UP=`date ; uptime`
echo "Uptime is $UP"
```

Upon execution, you will receive the following result –

```
Date is Thu Jul  2 03:59:57 MST 2009
Logged in user are 1
Uptime is Thu Jul  2 03:59:57 MST 2009
03:59:57 up 20 days, 14:03,  1 user,  load avg: 0.13, 0.07, 0.15
```

Variable Substitution

Variable substitution enables the shell programmer to manipulate the value of a variable based on its state.

Here is the following table for all the possible substitutions –

Sr.No.	Form & Description
1	**${var}** Substitute the value of *var*.
2	**${var:-word}** If *var* is null or unset, *word* is substituted for **var**. The value of *var* does not change.
3	**${var:=word}** If *var* is null or unset, *var* is set to the value of **word**.
4	**${var:?message}** If *var* is null or unset, *message* is printed to standard error. This checks that variables are set correctly.
5	**${var:+word}** If *var* is set, *word* is substituted for var. The value of *var* does not change.

Example

Following is the example to show various states of the above substitution –

Live Demo

```
#!/bin/sh

echo ${var:-"Variable is not set"}
echo "1 - Value of var is ${var}"

echo ${var:="Variable is not set"}
echo "2 - Value of var is ${var}"

unset var
echo ${var:+"This is default value"}
echo "3 - Value of var is $var"

var="Prefix"
```

```
echo ${var:+"This is default value"}
echo "4 - Value of var is $var"

echo ${var:?"Print this message"}
echo "5 - Value of var is ${var}"
```

Upon execution, you will receive the following result –

```
Variable is not set
1 - Value of var is
Variable is not set
2 - Value of var is Variable is not set

3 - Value of var is
This is default value
4 - Value of var is Prefix
Prefix
5 - Value of var is Prefix
```