UNIX - SHELL SUBSTITUTION

What is Substitution?

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

Example:

Following is the example, while printing value of the variable its substitued by its value. Same time "\n" is substituted by a new line:

```
#!/bin/sh

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

This would produce following result. Here **-e** option enables interpretation of backslash escapes.

```
Value of a is 10
```

Here is the result without -e option:

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

Here are following escape sequences which can be used in echo command:

Escape	Description
\	backslash
\a	alert (BEL)
\b	backspace
\c	suppress trailing newline
\f	form feed
\n	new line
\r	carriage return
\t	horizontal tab
\v	vertical tab

You can use **-E** option to disable interpretation of backslash escapes (default).

You can use **-n** option to disable insertion of 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 command substitution make sure that you are using 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 demonstrate command substitution:

```
#!/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"
```

This will produce 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:

Form	**Description**
${var}	Substitue the value of *var*.
${var:-word}	If *var* is null or unset, *word* is substituted for **var**. The value of *var* does not change.
${var:=word}	If *var* is null or unset, *var* is set to the value of **word**.
${var:?message}	If *var* is null or unset, *message* is printed to standard error. This checks that variables are set correctly.
${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:

```
#!/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}"
```

This would produce 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
```