UNIX / LINUX - SHELL QUOTING MECHANISMS

In this chapter, we will discuss in detail about the Shell quoting mechanisms. We will start by discussing the metacharacters.

The Metacharacters

Unix Shell provides various metacharacters which have special meaning while using them in any Shell Script and causes termination of a word unless quoted.

For example, **?** matches with a single character while listing files in a directory and an ***** matches more than one character. Here is a list of most of the shell special characters (also called metacharacters) –

```
* ? [ ] ' " \ $ ; & ( ) | ^ < > new-line space tab
```

A character may be quoted (i.e., made to stand for itself) by preceding it with a ****.

Example

Following example shows how to print a ***** or a **?** –

Live Demo

```
#!/bin/sh

echo Hello; Word
```

Upon execution, you will receive the following result –

```
Hello
./test.sh: line 2: Word: command not found

shell returned 127
```

Let us now try using a quoted character –

Live Demo

```
#!/bin/sh

echo Hello\; Word
```

Upon execution, you will receive the following result –

```
Hello; Word
```

The **$** sign is one of the metacharacters, so it must be quoted to avoid special handling by the shell –

Live Demo

```
#!/bin/sh

echo "I have \$1200"
```

Upon execution, you will receive the following result –

```
I have $1200
```

The following table lists the four forms of quoting –

Sr.No.	Quoting & Description
1	**Single quote** All special characters between these quotes lose their special meaning.
2	**Double quote** Most special characters between these quotes lose their special meaning with these exceptions – • $ • ` • \$ • \' • \" • \\
3	**Backslash** Any character immediately following the backslash loses its special meaning.
4	**Back quote** Anything in between back quotes would be treated as a command and would be executed.

The Single Quotes

Consider an echo command that contains many special shell characters –

```
echo <-$1500.**>; (update?) [y|n]
```

Putting a backslash in front of each special character is tedious and makes the line difficult to read –

```
echo \<-\$1500.\*\*\>\; \(update\?\) \[y\|n\]
```

There is an easy way to quote a large group of characters. Put a single quote (') at the beginning and at the end of the string –

```
echo '<-$1500.**>; (update?) [y|n]'
```

Characters within single quotes are quoted just as if a backslash is in front of each character. With this, the echo command displays in a proper way.

If a single quote appears within a string to be output, you should not put the whole string within single quotes instead you should precede that using a backslash (\) as follows –

```
echo 'It\'s Shell Programming'
```

The Double Quotes

Try to execute the following shell script. This shell script makes use of single quote –

Live Demo

```
VAR=ZARA
echo '$VAR owes <-$1500.**>; [ as of (`date +%m/%d`) ]'
```

Upon execution, you will receive the following result –

```
$VAR owes <-$1500.**>; [ as of (`date +%m/%d`) ]
```

This is not what had to be displayed. It is obvious that single quotes prevent variable substitution. If you want to substitute variable values and to make inverted commas work as expected, then you would need to put your commands in double quotes as follows –

Live Demo

```
VAR=ZARA
echo "$VAR owes <-\$1500.**>; [ as of (`date +%m/%d`) ]"
```

Upon execution, you will receive the following result –

```
ZARA owes <-$1500.**>; [ as of (07/02) ]
```

Double quotes take away the special meaning of all characters except the following –

- **$** for parameter substitution
- Backquotes for command substitution
- **\$** to enable literal dollar signs
- **\`** to enable literal backquotes
- **\"** to enable embedded double quotes
- **** to enable embedded backslashes
- All other **** characters are literal (not special)

Characters within single quotes are quoted just as if a backslash is in front of each character. This helps the echo command display properly.

If a single quote appears within a string to be output, you should not put the whole string within single quotes instead you should precede that using a backslash (\) as follows –

```
echo 'It\'s Shell Programming'
```

The Backquotes

Putting any Shell command in between **backquotes** executes the command.

Syntax

Here is the simple syntax to put any Shell **command** in between backquotes –

```
var=`command`
```

Example

The **date** command is executed in the following example and the produced result is stored in DATA variable.

Live Demo

```
DATE=`date`

echo "Current Date: $DATE"
```

Upon execution, you will receive the following result –

```
Current Date: Thu Jul  2 05:28:45 MST 2009
```