The echo Command

echo is a built-in *command* in the *bash* and *C shells* that writes its *arguments* to *standard output*.

A shell is a program that provides the command line (i.e., the all-text display user interface) on Linux and other Unix-likeoperating systems. It also executes (i.e., runs) commands that are typed into it and displays the results. bash is the default shell on Linux.

A command is an instruction telling a computer to do something. An argument is input data for a command. Standard output is the display screen by default, but it can be redirectedto a file, printer, etc.

The syntax for echo is

```
echo [option(s)] [string(s)]
```

The items in square brackets are optional. A *string* is any finite sequence of *characters* (i.e., letters, numerals, symbols and punctuation marks).

When used without any options or strings, echo returns a blank line on the display screen followed by the command prompt on the subsequent line. This is because pressing the ENTER key is a signal to the system to start a new line, and thus echo repeats this signal.

When one or more strings are provided as arguments, echo by default repeats those stings on the screen. Thus, for example, typing in the following and pressing the ENTER key would cause echo to repeat the phrase *This is a pen.* on the screen:

```
echo This is a pen.
```

It is not necessary to surround the strings with quotes, as it does not affect what is written on the screen. If quotes (either single or double) are used, they are not repeated on the screen.

Fortunately, echo can do more than merely repeat verbatim what follows it. That is, it can also show the value of a particular variable if the name of the variable is preceded directly (i.e., with no intervening spaces) by the dollar character ($), which tells the shell to substitute the value of the variable for its name.

For example, a variable named *x* can be created and its value set to 5 with the following command:

```
x=5
```

The value of x can subsequently be recalled by the following:

```
echo The number is $x.
```

Echo is particularly useful for showing the values of *environmental variables*, which tell the shell how to behave as a user works at the command line or in *scripts* (short programs).

For example, to see the value of HOME, the environmental value that shows the current user's home directory, the following would be used:

```
echo $HOME
```

Likewise, echo can be used to show a user's PATH environmental variable, which contains a colon-separated list of the directories that the system searches to find the executable program corresponding to a command issued by the user:

```
echo $PATH
```

echo, by default, follows any output with a *newline character*. This is a *non-printing* (i.e., invisible) character that represents the end of one line of text and the start of the next. It is represented by *\n* in Unix-like operating systems. The result is that the subsequent command prompt begins on a new line rather than on the same line as the output returned by echo.

The *-e* option is used to enable echo's interpretation of additional instances of the newline character as well as the interpretation of other special characters, such as a horizontal tab, which is represented by *\t*. Thus, for example, the following would produce a formatted output:

```
echo -e "\n Projects: \n\n\tplan \n\tcode \n\ttest\n"
```

(The above command should be written on a single line, although it may render as two lines on smaller display screens.) The *-n* option can be used to stop echo from adding the newline to output.

By making use of output redirection, echo provides a very simple way of creating a new file that contains text. This is accomplished by typing *echo* followed by the desired text, the output redirection operator (which is a rightward pointing angle bracket) and finally the name of the new file. The file can likewise be formatted by using special characters. Thus, for example, the formatted output from the above example could be used to create a new file called *project1*:

```
echo -e "\n Project1: \n\n\tplan \n\twrite \n\ttest\n" > project1
```

The contents of the new file, including any formatting, can be verified by using a command such as *cat* or *less*, i.e.,

```
less project1
```

echo can likewise be a convenient way of appending text to the end of a file by using it together with the the *append operator*, which is represented by two consecutive rightward pointing angle brackets. However, there is always the risk of accidentally using a single bracket instead of two, thereby overwriting all of the contents of the file, and thus, this feature is best reserved for use in scripts.

echo can also be used with *pattern matching*, such as the *wildcard character*, which is represented by the star character. For example, the following would return the phrase *The gif files are* followed by the names of all the .gif image files in the current directory:

```
echo -e The gif files are *.gif
```

echo is also commonly used to have a shell script display a message or instructions, such as *Please enter Y or N* in an interactive session with users.

echo is turned off automatically when passwords are entered so that they will not be shown on the screen.

Created September 19, 2005.