Shell Variables

Shell variables are used to hold working values within each nShell(tm) document. There are standard variables which are used by nShell to manage the processes of the shell. In addition, the user may define custom shell variables. Such variables provide a method of shorthand, often easing shell use.

Standard Variables

The standard shell variables are listed below:

TMP

- The directory used to hold temporary files
PATH
- The command search path
HOME
- The location of the nShell application
PWD
- The current working directory

The PATH variable may be modified using the "path"command. The "cd" command updates the PWD variable. In addition, the user may override any of these values using the "set" command.

By convention, upper case characters are used for shell control variables and lower case characters are used for user variables.

The "set", "unset" and "env" commands allow you to access shell variables from the command line.

Special Variables

The following variables serve special purposes within the nShell environment:

$?

The return value from the previous command
$#

The number of parameters passed to a script
$0

The name of an executing script

$1..$n

Parameters passed to a script

Using Shell Variables

set

The "set" command creates or modifies shell variables. Its format is "set <name> <value>". Variable names may be up to 31 characters in length and their values may be up to 255 characters. Consider the following command:

```
% set kitty "meow meow meow"
```

This sets the value of a variable called "kitty" to "meow meow meow".

env

The "env" (short for environment) command lists the contents of one or more shell variables:

```
% env kitty

kitty="meow meow meow"
```

An "env" command with no parameters lists all current shell variables.

unset

To remove an existing variable, use the "unset" command. Consider the sequence:

```
% set duck quack

% env duck

duck="quack"

% unset duck

% env duck

%
```

Examples

To evaluate or "expand" a shell variable within a command line, the "$" character is used. The shell looks up the name immediately following a "$" and substitutes its value:

```
% set duck quack

% echo $duck

quack
```

The variable expansion works within double quotes also:

```
% echo "Did I hear a $duck?"

Did I hear a quack?
```

Single quotes may be used when you want to prevent variable expansion:

```
% echo 'Did I hear a $duck?'
Did I hear a $duck?
```

Sometimes we get in a bind because we want to put the variable right up against other text. The following command fails because the shell reads it as asking for a variable called "ducking":

```
% echo "Did I hear $ducking?"
Did I hear $ducking?
```

The "" characters may be used to isolate a variable name:

```
% echo "Did I hear $ducking?"
Did I hear quacking?
```

Variables in Scripts

Shell variables may be used within scripts to manage passed parameters and track internal conditions. The "Scripting" section of this manual describes how shell parameters become the default shell variables $0..$n. In addition, the "set" "unset" and "env" commands may be used to create and modify variables within a script.

It is important to note that while a script may modify any variable, all changes are temporary and are discarded when the script terminates.