Here Scripts

Beginning with this lesson, we will construct a useful application. This application will produce an HTML document that contains information about our system. As we construct our script, we will discover step by step the tools needed to solve the problem at hand.

Writing an HTML File with a Script

As we may be aware, a well formed HTML file contains the following content:

```
<html>
<head>
    <title>
    The title of your page
    </title>
</head>

<body>
    Your page content goes here.
</body>
</html>
```

Now, with what we already know, we could write a script to produce the above content:

```
#!/bin/bash

# sysinfo_page - A script to produce an html file

echo "<html>"
echo "<head>"
echo "  <title>"
echo "  The title of your page"
echo "  </title>"
echo "</head>"
echo ""
echo "<body>"
echo "  Your page content goes here."
echo "</body>"
echo "</html>"
```

This script can be used as follows:

```
[me@linuxbox me]$ sysinfo_page > sysinfo_page.html
```

It has been said that the greatest programmers are also the laziest. They write programs to save themselves work. Likewise, when clever programmers write programs, they try to save themselves typing.

The first improvement to this script will be to replace the repeated use of the `echo` command with a single instance by using quotation more efficiently:

```
#!/bin/bash

# sysinfo_page - A script to produce an HTML file

echo "<html>
 <head>
   <title>
   The title of your page
   </title>
 </head>

 <body>
   Your page content goes here.
 </body>
 </html>"
```

Using quotation, it is possible to embed carriage returns in our text and have the `echo` command's argument span multiple lines.

While this is certainly an improvement, it does have a limitation. Since many types of markup used in HTML incorporate quotation marks themselves, it makes using a quoted string a little awkward. A quoted string can be used but each embedded quotation mark will need to be escaped with a backslash character.

In order to avoid the additional typing, we need to look for a better way to produce our text. Fortunately, the shell provides one. It's called a *here script*.

```
#!/bin/bash

# sysinfo_page - A script to produce an HTML file

cat << _EOF_
<html>
<head>
    <title>
    The title of your page
    </title>
</head>

<body>
    Your page content goes here.
</body>
</html>
_EOF_
```

A here script (also sometimes called a here document) is an additional form of I/O redirection. It provides a way to include content that will be given to the standard input of a command. In the case of the script above, the standard input of the `cat` command was given a stream of text from our script.

A here script is constructed like this:

```
command << token
content to be used as command's standard input
token
```

token can be any string of characters. "_EOF_" (EOF is short for "End Of File") is traditional, but we can use anything as long as it does not conflict with a bash reserved word. The token that ends the here script must exactly match the one that starts it, or else the remainder of our script will be interpreted as more standard input to the command which can lead to some really exciting script failures.

There is one additional trick that can be used with a here script. Often, we might want to indent the content portion of the here script to improve the readability of the script. We can do this if we change the script as follows:

```
#!/bin/bash

# sysinfo_page - A script to produce an HTML file

cat <<- _EOF_
    <html>
    <head>
        <title>
        The title of your page
        </title>
    </head>

    <body>
        Your page content goes here.
    </body>
    </html>
_EOF_
```

Changing the "<<" to "<<-" causes bash to ignore the leading tabs (but not spaces) in the here script. The output from the cat command will not contain any of the leading tab characters. This technique is a bit problematic, as many text editors are configured (and desirably so) to use sequences of spaces rather than tab characters.

O.k., let's make our page. We will edit our page to get it to say something:

```
#!/bin/bash

# sysinfo_page - A script to produce an HTML file

cat <<- _EOF_
    <html>
    <head>
        <title>
        My System Information
        </title>
    </head>

    <body>
    <h1>My System Information</h1>
    </body>
    </html>
_EOF_
```

In our next lesson, we will make our script produce some real information about the system.