Command Substitution And Constants

In the previous lesson, we learned how to create variables and perform expansions with them. In this lesson, we will extend this idea to show how we can substitute the results from a command.

When we last left our script, it could create an HTML page that contained a few simple lines of text, including the host name of the machine which we obtained from the environment variable HOSTNAME. Now, we will add a time stamp to the page to indicate when it was last updated, along with the user that did it.

```
#!/bin/bash

# sysinfo_page - A script to produce an HTML file

title="System Information for"

cat <<- _EOF_
    <html>
    <head>
        <title>
        $title $HOSTNAME
        </title>
    </head>

    <body>
    <h1>$title $HOSTNAME</h1>
    <p>Updated on $(date +"%x %r %Z") by $USER</p>
    </body>
    </html>
_EOF_
```

As you can see, we employed another environment variable, `USER`, to get the user name. In addition, we used this strange looking thing:

```
$(date +"%x %r %Z")
```

The characters "$( )" tell the shell, "substitute the results of the enclosed command." In our script, we want the shell to insert the results of the command `date +"%x %r %Z"` which expresses the current date and time. The **date** command has many features and formatting options. To look at them all, try this:

```
[me@linuxbox me]$ date --help | less
```

Be aware that there is an older, alternate syntax for "$(command)" that uses the backtick character " ` ". This older form is compatible with the original Bourne shell (sh). I tend not to use the older form since I am teaching modern `bash` here, not `sh`, and besides, I think backticks are ugly. The bash shell fully supports scripts written for sh, so the following forms are equivalent:

```
$(command)
`command`
```

Assigning A Command's Result To A Variable

You can also assign the results of a command to a variable:

```
right_now=$(date +"%x %r %Z")
```

You can even nest the variables (place one inside another), like this:

```
right_now=$(date +"%x %r %Z")
time_stamp="Updated on $right_now by $USER"
```

Constants

As the name variable suggests, the content of a variable is subject to change. This means that it is expected that during the execution of your script, a variable may have its content modified by something you do.

On the other hand, there may be values that, once set, should never be changed. These are called *constants*. I bring this up because it is a common idea in programming. Most programming languages have special facilities to support values that are not allowed to change. Bash also has these facilities but, to be honest, I never see it used. Instead, if a value is intended to be a constant, it is given an uppercase name to remind the programmer that it should be considered a constant even if it's not being enforced.

Environment variables are usually considered constants since they are rarely changed. Like constants, environment variables are given uppercase names by convention. In the scripts that follow, I will use this convention - uppercase names for constants and lowercase names for variables.

So with everything we know, our program looks like this:

```
#!/bin/bash

# sysinfo_page - A script to produce an HTML file

title="System Information for $HOSTNAME"
```

```
title "System Information for $HOSTNAME"
RIGHT_NOW=$(date +"%x %r %Z")
TIME_STAMP="Updated on $RIGHT_NOW by $USER"

cat <<- _EOF_
    <html>
    <head>
        <title>
        $title
        </title>
    </head>

    <body>
    <h1>$title</h1>
    <p>$TIME_STAMP</p>
    </body>
    </html>
_EOF_
```
