# Variables

```
#!/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_
```

Now that we have our script working, let's improve it. First off, we'll make some changes because we want to be lazy. In the script above, we see that the phrase "My System Information" is repeated. This is wasted typing (and extra work!) so we'll improve it like this:

```
#!/bin/bash

# sysinfo_page - A script to produce an HTML file

title="My System Information"

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

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

We added a line to the beginning of the script and replaced the two occurrences

of the phrase "My System Information" with `$title`.

## Variables

What we have done is to introduce a fundamental concept that appears in every programming language, *variables*. Variables are areas of memory that can be used to store information and are referred to by a name. In the case of our script, we created a variable called "title" and placed the phrase "My System Information" into memory. Inside the here script that contains our HTML, we use "$title" to tell the shell to perform *parameter expansion* and replace the name of the variable with the variable's contents.

Whenever the shell sees a word that begins with a "$", it tries to find out what was assigned to the variable and substitutes it.

## How to Create a Variable

To create a variable, put a line in the script that contains the name of the variable followed immediately by an equal sign ("="). No spaces are allowed. After the equal sign, assign the information to store.

## Where Do Variable Names Come From?

We just make them up. That's right; we get to choose the names for our variables. There are a few rules.

1. Names must start with a letter.
2. A name must not contain embedded spaces. Use underscores instead.
3. Punctuation marks are not permitted.

## How Does This Increase Our Laziness?

The addition of the `title` variable made our life easier in two ways. First, it reduced the amount of typing we had to do. Second and more importantly, it made our script easier to maintain.

As we write more and more scripts (or do any other kind of programming), we will see that programs are rarely ever finished. They are modified and improved by their creators and others. After all, that's what open source development is all about. Let's say that we wanted to change the phrase "My System Information" to "Linuxbox System Information." In the previous version of the script, we would have had to change this in two locations. In the new version with the `title` variable, we only have to change it in one place. Since our script is so small, this might seem like a trivial matter, but as scripts get larger and more complicated, it becomes very important.

## Environment Variables

When we start our shell session, some variables are already set by the startup files we looked at earlier. To see all the variables that are in the environment, use the **`printenv`** command. One variable in our environment contains the host name for the system. We will add this variable to our script like so:

```
#!/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>
    </body>
    </html>
_EOF_
```

Now our script will always include the name of the machine on which we are running. Note that, by convention, environment variables names are uppercase.

---