In Bash you can store data in variables. In chapter 4 we discussed environmental variables that are set by your operating system. You can also create your own variables. Make sure you follow these rules when you're naming variables:

- Every character should be lowercase.
- The variable name should start with a letter.
- The name should only contain alphanumeric characters and underscores (_).
- Words in the name should be separated by underscores.

If you follow those rules then you can avoid accidentally overwriting data stored in environmental variables.

You can assign data to a variable using the equals sign (=). The data you store in a variable can either be a string or a number. Let's create a variable now on the command line:

```
chapter_number=5
```

The variable name is on the left hand side of the equals sign, and the data which will be stored in that variable is on the right hand side of the equals sign. Notice that there are no spaces on either side of the equals sign, this is not allowed when assigning variables:

```
chapter_number = 5

## Error in running command bash
```

In order to print the data in a variable, also called the value of a variable, we can use echo. When you want to retrieve the value of a variable you must use the dollar sign ($) before the name of the variable. Let's try this out:

```
echo $chapter_number

## 5
```

You can modify the value of a variable using arithmetic operators by using the let command:

```
let chapter_number=$chapter_number+1
echo $chapter_number

## 6
```

You can also store strings in variables:

```
the_empire_state="New York"
echo $the_empire_state

## New York
```

Occasionally you might want to run a command like you would on the command line and store the result of that command in a variable. We can do this by wrapping the command in a dollar sign and parentheses ($( )) around a command. This syntax is called command substitution. The command is executed and then gets replaced by the string that resulted from running the command. For example if we wanted to store the number of lines in math.sh:

```
math_lines=$(cat math.sh | wc -l)
echo $math_lines

## 7
```

Variable names with a dollar sign can also be used inside other strings in order to insert the value of the variable into the string:

```
echo "I went to school in $the_empire_state."

## I went to school in New York.
```

When writing a Bash script, the script gives you a few variables for free. Let's create a new file called vars.sh with the following code:

```
#!/usr/bin/env bash
# File: vars.sh

echo "Script arguments: $@"
echo "First arg: $1. Second arg: $2."
echo "Number of arguments: $#"

```

Now let's try running the script a few times in a few different ways:

```
bash vars.sh

## Script arguments:
## First arg: . Second arg: .
## Number of arguments: 0

bash vars.sh red

## Script arguments: red
## First arg: red. Second arg: .
## Number of arguments: 1

bash vars.sh red blue

## Script arguments: red blue
## First arg: red. Second arg: blue.
## Number of arguments: 2

bash vars.sh red blue green

## Script arguments: red blue green
## First arg: red. Second arg: blue.
## Number of arguments: 3
```

Your script can accept arguments just like a command line program! The first argument to your script is stored in $1, the second argument is stored in $2, etc, etc. An array of all of the arguments passed to your script is stored in $@, and we'll discuss how to handle arrays later on in this chapter. The total number of arguments passed to your script is stored in $#. Now that you know how to pass arguments to your scripts you can start writing your own command line tools!

Summary

- Variables can be assigned with the equal sign (=) operator.
- Strings or numbers can be assigned to variables.
- The value of a variable can be accessed with the dollar sign ($) before the variable name.
- You can use the dollar sign and parentheses syntax (command substitution) to execute a command and save the output in a variable.
- You can access command line arguments within your own scripts using the dollar sign followed by the number of the argument.

Mark as completed