Writing a Shell Script

To successfully write a shell script, you have to do three things:

```
1.Write a script
2.Give the shell permission to execute it
3.Put it somewhere the shell can find it
```

Writing A Script

A shell script is a file that contains ASCII text. To create a shell script, just use a text editor. Here is an example of a simple Shell Script:

```
#!/bin/bash
# My first script
echo "Hello World!"
Now save the file as helloworld
```

The first line of the script is important. This is a special clue, called a shebang, given to the shell indicating what program is used to interpret the script. In this case, it is /bin/bash. Other scripting languages such as Perl, awk, tcl, Tk, and python also use this mechanism.

The second line is a comment. Everything that appears after a “#” symbol is ignored by bash. As your scripts become bigger and more complicated, comments become vital. They are used by programmers to explain what is going on so that others can figure it out. The last line is the echo command. This command simply prints its arguments on the display.

Setting Permissions

The next thing we have to do is give the shell permission to execute your script. This is done with the chmod command as follows:

```
chmod 755 helloworld (sudo if you are using Raspbian)
```

The “755” will give you read, write, and execute permission. Everybody else will get only read and execute permission. If you want your script to be private (i.e., only you can read and execute), use “700” instead.

At this point, your script will run. Try this:

```
./hello_world
```

You should see “Hello World!” displayed. If you do not, see what directory you really saved your script in, go there and try again.

Paths

Before we go any further, I have to stop and talk a while about paths. When you type in the name of a command, the system does not search the entire computer to find where the program is located. That would take a long time. You have noticed that you don't usually have to specify a complete path name to the program you want to run, the shell just seems to know.

Well, you are right. The shell does know. Here's how: the shell maintains a list of directories where executable files (programs) are kept, and only searches the directories in that list. If it does not find the program after searching each directory in the list, it will issue the famous command not found error message.

This list of directories is called your path. You can view the list of directories with the following command:

```
echo $PATH
```

This will return a colon separated list of directories that will be searched if a specific path name is not given when a command is attempted. In our first attempt to execute your new script, we specified a pathname (“./”) to the file.

You can add directories to your path with the following command, where directory is the name of the directory you want to add:

```
export PATH=$PATH:directory
```

A better way would be to edit your .bash_profile or .profile file (depending on your distribution) to include the above command. That way, it would be done automatically every time you log in.

Most Linux distributions encourage a practice in which each user has a specific directory for the programs he/she personally uses. This directory is called bin and is a subdirectory of your home directory. If you do not already have one, create it with the following command:

```
mkdir bin
```

Move your script into your new bin directory and you're all set. Now you just have to type:

```
hello_world
```

Now your script will run. On some distributions, most notably Ubuntu, you will need to open a new terminal session before your newly created bin directory will be recognised.

This content came from:

```
http://linuxcommand.org/lc3_wss0010.php
```