Code Matters

Modulinos In Bash

June 12, 2021

A modulino is a file which behaves like a library when it is imported, and like a script when executed. I first read about them in Mastering Perl, but you can create them in other languages too. Here's how to do it in Bash.

Let's say you have a simple script:

```
#!/bin/bash

echo "Hello, World!"
```

A common refactor with scripts is to encapsulate all the code behavior in functions. As this script only does one thing, it's a small change:

```
#!/bin/bash

function hello {
  echo "Hello, World!"
}
hello
```

Now instead of printing "Hello, World!", I can make it greet whatever name it is given instead.

```
#!/bin/bash

function hello {
  if [[ -n "$1" ]];then
    name="$1"
  else
    name="World"
  fi
  echo "Hello, $name!"
}
hello "$@"
```

The line `hello "$@"` calls the hello function, passing any command line arguments the script received. I've added an `if` clause to use the first function argument if it's not empty, else to default to "World!" and preserve the original behavior.

To make this script behave like a library, I just add a condition to the `hello` function call:

```
#!/bin/bash

function hello {
  if [[ -n "$1" ]];then
    name="$1"
  else
    name="World"
  fi
```

```
  echo "Hello, $name!"
}
[[ "$BASH_SOURCE" == "$0" ]] && hello "$@"
```

This checks that the BASH_SOURCE variable (the script name) equals $0 which is the name of the file invoking the code. These values match when the script is invoked like a program: `./hello.bash`. But when the script is *sourced* `$0` evaluates to `/bin/bash`.

This can be useful if you'd like to be able to import your script code into other scripts via `source`. One reason to do that is unit testing your scripts:

```
#!/bin/bash

PASS=0

function fail {
  echo "$1"
  PASS=1
}

source "./hello.bash"

def=$(hello)
[[ "$def" == "Hello, World!" ]] || fail "wrong default greeting: $def"

arg=$(hello David)
[[ "$arg" == "Hello, David!" ]] || fail "wrong arg greeting: $arg"

exit "$PASS"
```

Tags: bash shell library

Follow me on Twitter @perltricks