5. Variables

You can use variables as in any programming languages. There are no data types. A variable in bash can contain a number, a character, a string of characters.

You have no need to declare a variable, just assigning a value to its reference will create it.

5.1 Sample: Hello World! using variables

```
#!/bin/bash
STR="Hello World!"
echo $STR
```

Line 2 creates a variable called STR and assigns the string "Hello World!" to it. Then the VALUE of this variable is retrieved by putting the '$' in at the beginning. Please notice (try it!) that if you don't use the '$' sign, the output of the program will be different, and probably not what you want it to be.

5.2 Sample: A very simple backup script (little bit better)

```
#!/bin/bash
OF=/var/my-backup-$(date +%Y%m%d).tgz
tar -cZf $OF /home/me/
```

This script introduces another thing. First of all, you should be familiarized with the variable creation and assignation on line 2. Notice the expression '$(date +%Y%m%d)'. If you run the script you'll notice that it runs the command inside the parenthesis, capturing its output.

Notice that in this script, the output filename will be different every day, due to the format switch to the date command(+%Y%m%d). You can change this by specifying a different format.

Some more examples:

echo ls

echo $(ls)

5.3 Local variables

Local variables can be created by using the keyword *local*.

```
#!/bin/bash
HELLO=Hello
function hello {
        local HELLO=World
        echo $HELLO
}
echo $HELLO
hello
echo $HELLO
```

This example should be enought to show how to use a local variable.

Next Previous Contents