IT Scripting and Automation

Scripting in Linux
-Variables-

Lecturer: Art Ó Coileáin

Scripts

- A script contains a series of commands.
- An interpreter will execute the commands in the script one after the other.
- What can be in a shell script?
  - Anything that can be typed at the command line.
- What is the purpose of a shell script?
  - To automate tasks.

Scripts

Defining the Interpreter

- An interpreter will execute the commands in the script one after the other.
- It is good practice to specify the type of interpreter we want to use in the first line of a script. This line is called a bang line or more commonly a shebang. It indicates to the system how we want this file to be executed.
- Our shebang starts with a hash sign and exclamation mark, followed by the absolute path to the bash interpreter. E.g.:

```
#!/bin/bash
echo “Hello World ! This is my first script.”
```

Script: *myFirstScript.sh*

- Execution of the script:

```
chmod 755 myFirstScript.sh

./myFirstScript.sh

Hello World ! This is my first script
```

Script: *sleepScript.sh*

Another example:

```
#!/bin/bash
sleep 100
```

```
./sleepScript.sh &
[1] 965
```

* this number may be different in your system and per run

```
$ ps -fp 965
UID      PID    PPID   C       STIME   TTY   TIME       CMD
Student  965    758    0       11:10   tty         00:00:00   sleep 100
```

Variables in Shell Scripts

Variables

- Variables are an important part of any programming language, and Bash is no different.
- When you start a new session from the terminal, the shell already sets some variables for you.
- We call these environment variables, because they usually define characteristics of our shell environment.

Variables in Shell Scripts

Points to Remember:

- Variables must contain only alphanumeric characters or underscores.
- Variables are case sensitive (in Linux/Unix)
- Variables in Bash have an *implicit type*, and are considered strings.
- **By convention variables are uppercase**
- Syntax

  ```
  VARIABLE_NAME="value"
  ```

- Make sure you don't use spaces before or after the equal sign.

Variables Definition - Examples

- MY_MESSAGE=”Hello, this is a message”
- ANOTHER_VARIABLE=”This is the value”
- MY_NUMBER=32
- MY_LUCKY_NUMBER=1
- MIXED_VARIABLE=123abc

- **Incorrect Variable Definition:**
- `MY MESSAGE=”Hello”`
- `ANOTHER_VARIABLE = “This is the value”`
- `3CHARACTERS=“abc”`

Variables in Shell Scripts

Script Usage:

- To use a variable put **$** before the variable name

```
echo “ Example of variable: $VARIALBE_NAME ”
```

Variables in Shell Scripts, Examples:

```
#!/bin/bash
My_FIRST_VARIABLE=“Scripting”
echo “ $My_FIRST_VARIABLE is cool”
```

```
#!/bin/bash
My_FIRST_VARIABLE=“Scripting”
echo “ ${My_FIRST_VARIABLE}is cool”
```

NOTE: if **‘{‘** and **‘}’** are used then no space is needed **“…}is…”**

A command output into a variable

- #!/bin/bash
- NAME=$(hostname)
- echo “The name of my server is: ${NAME}.”

Execution:

./scriptVariables2.sh

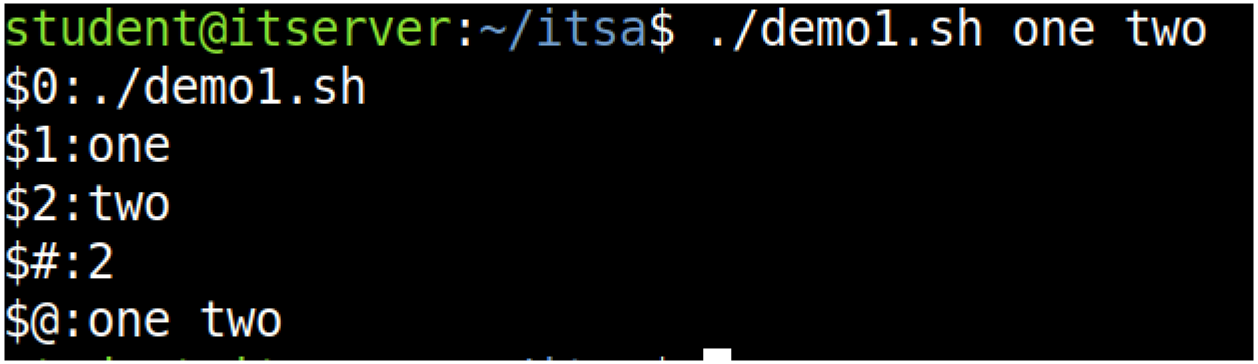
Output:

The name of my server is: ITSA-Server

Script Arguments/Positional Parameters

You are already familiar with using arguments in the Linux core utilities. e.g.: `rm testfile` contains both the executable rm and one argument testfile.

- The arguments can be passed to scripts upon execution, and are easily accessible as positional variables: **$1**, **$2**….
- **$0** contains the name of the invoked script
- **$#** contains a count of the arguments passed to the script
- **$@** contains the full array of all positional parameters.



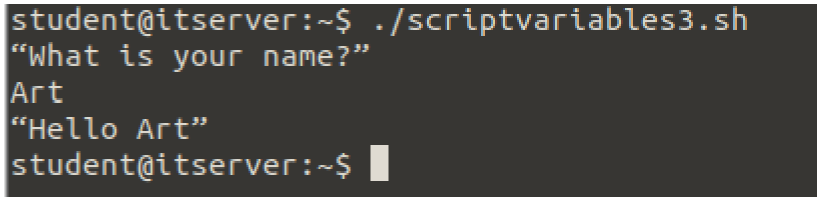
```
student@itserver:~/itsa$ ./demo1.sh one two
$0:./demo1.sh
$1:one
$2:two
$#:2
$@:one two
```

How to read from the standard input?

```
#!/bin/bash
MY_NAME=“”
echo “What is your name?”
read MY_NAME
echo “Hello $MY_NAME”
```

Execution:

./scriptVariables3.sh



```
student@itserver:~$ ./scriptvariables3.sh
“What is your name?”
Art
“Hello Art”
student@itserver:~$
```

Exercise

Write a script to create empty files and list them after their creation:

1. The filenames should be starting with a student name and ending with the following extensions: “old”, “bckup1” and “bckup2”
2. Use a variable for the name of the student

```
./generateFiles.sh
```

Expected output:

```
What is your name? John
Hello John, Your files (Johnold, Johnbckup1 and Johnbcup2) have…
…
Johnold
Johnbckup1
Johnbckup2
```