IT Security course
-
SHL 1

Sylvain Leroy
sylvain@eternilab.com

November 8, 2019

Contents

Chapter 1

SHL 1

1.1 License

1.2 Introduction

Scripting let you code programs easily and are very flexible. Scripts are totally architecture independent (exceptions may exist) and are a very good way to automatise actions on a system.

The purpose of this course is to teach you *basic tools* and vital *rigor* to have when automatising tasks on a UNIX-like system.

Language to use : Shell (bash)

1.3 Instructions

Here is what you MUST respect in order to do your work:

1. If a list of function/tool is given at the beginning of the exercise, you can ONLY use these for that exercise.
2. If no function/tool name is listed at the beginning of an exercise, you can use any function/tool you want for that exercise.
3. The sign $ is the prompt of your shell. The line where you send inputs.
4. You MUST give back a tarball using template name: *your_lastname*.tar.bz2
5. In case of doubt, you MUST send an e-mail to ask a question on what you doubt about. If no question are asked by anyone, the decision will remain to the teacher.
6. Every e-mail you send must have the object starting with [SHL1][X], where X can be : QUESTION or HANDIN. Every other e-mail will be discarded.

The tarball you'll be giving back must *EXACTLY* look like this (using *tree* command):

```
$ tree your_lastname/
your_lastname
|-- ANSWERS
|-- AUTHORS
|-- ex1
|   --- friends.sh
|-- ex2
|   --- tree.sh
|-- ex3
|   --- alphanum.sh
|-- ex4
|   --- mix.sh
|-- ex5
|   --- tower.sh
|-- ex6
|   --- rotation.sh
|-- ex7
|   --- checksum.sh
--- README

7 directories, 10 files
```

- ANSWERS contains your answers (if any).
- AUTHORS contains you 'firstname lastname' (in that order and followed by an end of line) and the names of your mates if you work in team (one name per line).
- README contains nothing (file must be empty).

1.4 Using shell

Let's start!

Question: Name 3 existing scripting languages.
Question: Name 3 existing shells.

1.4.1 SheBang

A SheBang is the first line in a shell script (in scripts in general in fact).
It calls the binary of the targeting shell program.

Good way

Historicaly, you would have used this shebang:

```
#! /bin/sh

echo "foo"
```

Best way

If your script is targeting userspace (with /usr filesystem mounted), then you should use this shebang:

```
#! /usr/bin/env sh

echo "foo"
```

That one makes your program portable over Unix systems (every of them where *env* binary exists).
env look for the right path of the shell called as argument 1 (here sh) and starts it.

1.4.2 Outputs

Strings outputs

When writing shell scripts, you can write on 2 different outputs:

- STDOUT
- STDERR

The first one is used for standard information and the second one is used to log errors.

Return value

The return value is the exit status code of your program.

When your program is called from another program, that other program may not be able to read your standard output and will most of the time try to know if your program did the right stuff or couldn't make it using its return value.

Return values are kind of standardized:

- 0 means OK, the program ran smoothly

- negatives values means errors. If the return value is an unsigned number, any value other than 0 means error.

1.4.3 debugging

To debug your scripts, you can use at the begining of your scripts :

- set -x
  Enable debug lines printing
- set -e
  Enable stop on error

1.4.4 Builtins

Shells often implement functions named builtins. They are tiny but useful commands.

Question: Name 3 builtins from the shell *Bash*.

1.4.5 Shell special parameters

Question: Name the 3 following special parameters and tell what they are used for:

- $0
- $?
- $#

1.4.6 man

man is the best command ever. Learn to use it:

```
$ man man
```

Here is a tiny summary on its command :

- / + string
  Exemple : /toto - will search the keywork *toto* in the man page
- n
  Will go to the next search result
- N
  Will go to the previous search result
- q
  Will exit the man page

If you prefer the *info* command, you're welcome, but keep in mind that many programs don't have any info pages (the opposite is also true).

1.5 Basic tools

Here are lists of tools you need to know and read the manual pages.

1.5.1 Base Manipulation

- ls
- touch
- rm
- ps
- kill
- file

1.5.2 Disk

- df
- du

1.5.3 Network

- ifconfig
- ip

1.5.4 Monitoring

- top
- htop
- watch
- vmstat

1.5.5 Accounting

- sa

1.6 Exercise 1: Friends

You have to write a shell script that takes a list of friends as argument and writes on the standard output the number of them and their names.

Your script must return the number of friend(s).

Examples:

```
$ ./friends.sh
You have no friend.
$ echo $?
0
$ ./friends.sh foo
You have 1 friend: foo.
$ echo $?
1
$ ./friends.sh foo bar
You have 2 friends: foo, bar.
$ echo $?
2
```

1.7 Exercise 2: Tree

You have to write a script that takes a number as argument and prints a tree of the corresponding size. The tree trunk size is the half of the size of the tree (rounded down).

Your script must return 0.

Examples:

```
$ ./tree.sh
$ ./tree.sh 0
Too tiny.
$ ./tree.sh 1
Too tiny.
$ ./tree.sh 2
 *
***
 |
$ ./tree.sh 3
  *
 ***
*****
  |
$ ./tree.sh 5
    *
   ***
  *****
 *******
*********
    |
    |
```

1.8 Exercise 3: Alphanum

Your have to write a script that reads one or many strings from standard input and says if this an alpha string, an alphanumeric string or only a numeric string.

Your script must return 0.

Examples:

```
$ ./alphanum.sh
sqlqsj
It's a word.
4329562
It's a number.

It's empty.
3245azrtd44
It's a mix.
u\^o*wx
Wait... What?
$ echo $?
0
```

1.9 Exercise 4: Mix

You have to write a script that takes 2 files in argument and prints each line of the files one after the other.

Your script must return the number of line of the smallest file.

Examples:

```
$ cat file1
a
b
c
$ cat file2
1
2
3
$ ./mix.sh file1 file2
a
1
b
2
c
3
$ echo $?
3
```

1.10 Exercise 5: Tower

You have to write a script that draw in ASCII-art a tower on the STDERR.

It takes in argument the number of floor and the type of windows of the tower

Your script must return:

- 0 if everything is OK
- 1 on error

Examples:

```
$ ./tower.sh
$echo $?
1
$ ./tower.sh 2 square
 -----------------
|  _ _       _ _  |
| |_|_|     |_|_| |
| |_|_|     |_|_| |
|                 |
|  _ _       _ _  |
| |_|_|     |_|_| |
| |_|_|     |_|_| |
|                 |
|        _        |
|       | |       |
|_______| |_______|
$echo $?
0
$ ./tower.sh 1 triangle
 -----------------
|                 |
|  / \      / \   |
| /_ _\    /_ _\  |
|                 |
|        _        |
|       | |       |
|_______| |_______|
$echo $?
0
```

1.11 Exercise 6: Rotation

Your have to write a scripts that takes a string and a number. The number indicates how many time every letter in the string must rotate (in alphabetic order).

Your script must return the number of letter that where rotated.

Examples:

```
$ ./rotation.sh abcdef 1
bcdefg
$ ./rotation.sh def..gcd 2
fgh..ief
```

1.12 Exercise 7: Checksum

Your have to write a script that takes a path in argument, computes a md5 sum, a sha256 sum and a sha512 sum, then prints it on the standard output using pretty printing like that:

```
Filename:
md5 is md5checksum
sha256 is sha256checksum
sha512 is sha512checksum
```

The script must return the number of file checksummed.

Examples:

```
$ ./checksum.sh .
AUTHORS:
md5 is d41d8cd98f00b204e9800998ecf8427e
sha256 is e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
sha512 is cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d85f2b0ff8
README:
md5 is d41d8cd98f00b204e9800998ecf8427e
sha256 is e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
sha512 is cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d85f2b0ff8
ex1/friends.sh:
md5 is d41d8cd98f00b204e9800998ecf8427e
sha256 is e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
sha512 is cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d85f2b0ff8
```

1.13 Bibliography

Advanced Bash-Scripting Guide : http://tldp.org/LDP/abs/html/