UNIVERSITÉ DU LUXEMBOURG

Operating Systems 1

Christian Grévisse (christian.grevisse@uni.lu)
Johannes Klein (johannes.klein@uni.lu)

Lab 7 – Bash Scripts

2017-12-05

Remember that you need to assign execution rights to your scripts in order to run them: `chmod u+x myScript.sh`

Exercise 1 – Greetings!

Using the command `whoami`, write a bash script that greets the current user.

Exercise 2 – Shell for Dummies

Using the `select` and `case` constructs, write a script for inexperienced users that allows them to repeatedly perform one of the following tasks

- list the contents of the current working directory
- change the working directory
- create a new (empty) file in the working directory
- create a new subdirectory in the working directory
- remove a file
- remove a directory
- ... any other command you want to offer

without having to type any commands themselves. The choice of the task and the path of a file or directory are the only input required by users.

Exercise 3 – Absolute Paths

Using command substitution, write a script that takes a directory path as command line argument and displays the absolute paths of all of its entries. You will probably need the commands `cd`, `ls` and `pwd`. Make sure that the user input is correct, i.e. that a path is indicated as a command line argument and that the indicated path is indeed a directory. Also, make sure that the output is correct for all cases, especially consider the root of the file system `/`.

> ✎ **Absolute Paths vs. Relative Paths**
>
> An **absolute path** indicates the full path to a file system entry and is characterized by a leading `/`. For instance, `/home/student/Desktop/A` is the absolute path to some directory `A` on the `Desktop`.
>
> A **relative path** is the path of a file system entry relative to some directory. For instance, `Desktop/A` or `./Desktop/A` is the path to the same directory `A` relative to the home folder.

Exercise 4 – Leap Year (`if-elif-else`, command line arguments, `echo` & `read`)

Write a script that checks whether a given year is a leap year. If a command line argument is given, it is considered as the year to be checked. Otherwise, the user shall be prompted to enter a year.

Exercise 5 – Sum (command line arguments, `shift`, loops, variable manipulation, `case`)

Write a script that takes an arbitrary number of numbers as command line arguments and displays their sum. Implement the sum in 3 different ways, namely with a `for`-loop, a `while`-loop and an `until`-loop. In the first command line argument, the user chooses between those 3 calculation ways.

Exercise 6 – File & Directory Lists (`for`-loop, globbing, variable manipulation, escaped characters)

Write a script that takes the path to a directory as command line argument and prints two different lists, a list of all files contained in the directory and a list of all subdirectories. Verify that the given path is indeed the path to a directory.

The output should look like this:

```
---- Directories (3) ----
Some Directory
Some Other Directory
Yet Another Directory

---- Files (2) ----
file1
file2
```

Exercise 7 – File Size

Write a script that takes the path to a file (sic!) as command line argument and prints the file size (and nothing but the file size, i.e. no trailing spaces or anything else than the number). The `ls` command might be useful.

Exercise 8 – Time to say Hello!

The `date` command prints the system date and time:

```
$ date
Tue Dec  6 12:34:56 CET 2016
```

Write a script that shows *"Good morning"* before 12:00, *"Good afternoon"* between 12:00 and 17:00 and *"Good evening"* after 17:00.

Exercise 9 – File Playground

Write a script that takes 4 command line arguments:

1. the path to a file f
2. an old file extension e_1, e.g. `.sh`
3. a new file extension e_2, e.g. `.txt`
4. the path to a directory d

We assume that file f contains, in each line, the absolute path to some file. To retrieve these paths, you can use a `for`-loop over the content of f (*hint:* use `cat`). For each file, we first check its existence and whether it matches the file extension e_1. To do so, you may use the `basename` command, which can extract the file's name without extension. Matching files will be copied with the new extension e_2 into directory d. If d does not exist, create it from within the script!

To test your script, use the output of the script from exercise 3.

Exercise 10 – Rights on programs

Write a script that takes the name of a program (e.g. some command) as a command line argument and outputs the rights the user executing the script has on this program. *Hint:* Use `which` to determine the path of the program.