Exercise 6.1

The fact that subshells forget certain things once they are left, is not

only a pain, but can be really useful as well. A typically example is if one wants to do a

particular task for all subdirectories of a particular directory.

In this exercise, we want to design a script, which prints the name of the largest file

for each subdirectory of the resources directory of the bash course.

There are many ways to do this. For the sake of the exercise do not use an external

program like find to traverse the directory tree, but instead really cd into a directory

first, before finding the largest file in it.

A few hints:

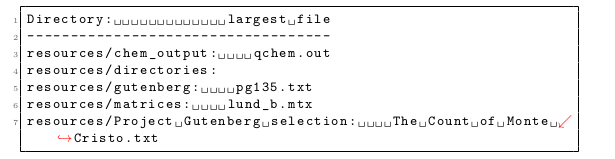
• For now there is no need to recurse, i.e. just go into all immediate subdirectories

of resources, find the largest file and print it. No need to look at subdirectories

of subdirectories . . .

• If multiple files have the same size, just print one of them for simplicity.

• The result could look something like:

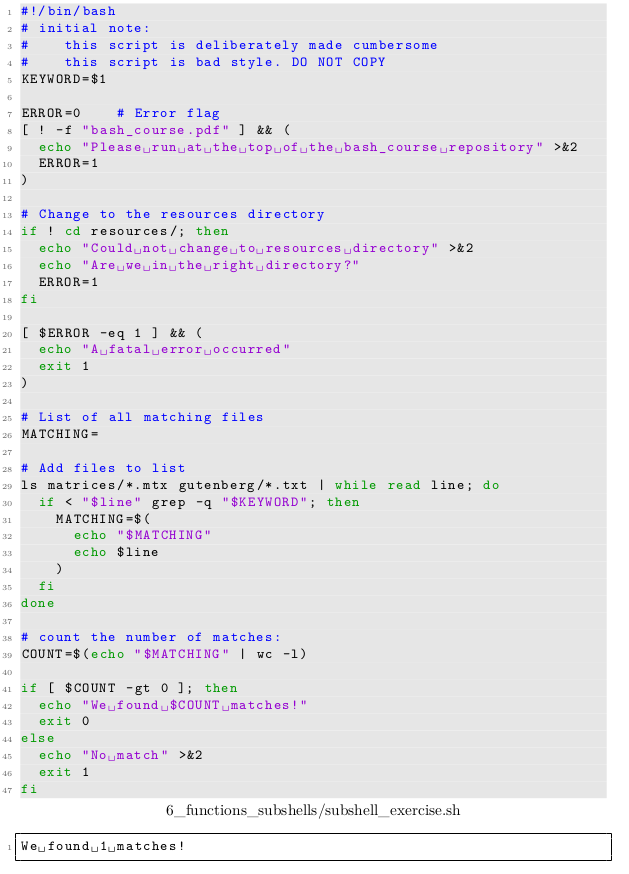


• A very helpful commands for this exercise is wc. You may use ls as well, but if

you think the wrong way, the exercise can become complicated.

Exercise 6.2.

This (found in 6\_functions\_subshells/subshell\_exercise.sh in the git directory) script does not produce the results the author expected. Spot the errors and correct them. You should find roughly 3 problems.



Exercise 6.3

. Rebuild the find -type f command using only the features of the bash shell. That is your script should list the relative path to all files in all subdirectories and subsubdirs . . . of the current working dir. Some hints:

• Do not worry about the full task at first. Imagine your working directory is a

particular directory, resources say. In this directory you will find other directories

and of course files. Only deal with the files for now, i.e.: Write a bash function,

which lists all files within a directory.

• The for file in \*; do-loop is your friend here.

• Extend the above function such that it calls itself to process the immediate subdi-

rectories as well. This strategy to solve this problem is called recursive processing.

• Now try to achieve the full goal. Use subshells to keep track of the current directory

level you are in and be careful to print really the full path to a particular file like

find -type f does it as well.

Exercise 6.4

Take another look at your script from the second Project Gutenberg

exercise (exercise 5.4). Split the script into a few sensible functions. Some

Ideas:

• Have one function to parse read the tabular output of exercise 4.6 and compute the ξ

numbers. The results could be sent to stdout in another tabular form which shows

the ξ numbers and the file:



• One function to read the list produced above and print three recommended books

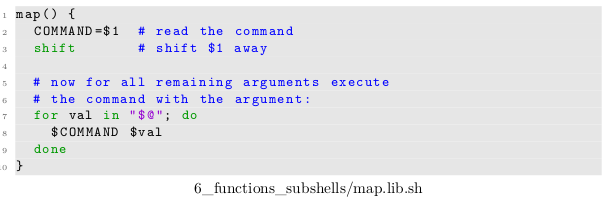
to stdout

• The main body should just call the example 4.6 script and use the functions defined

above to process what the ex-4.6-script yields.

Exercise 6.5

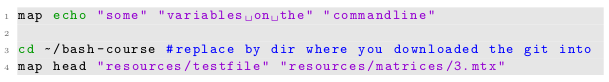
In this exercise we will try some abstract bash programming using functions. First take a look at the following code:



It defines a so-called mapping function, which applies a command or a function name to

all arguments provided in turn. Copy the code to a fresh file and add the following lines

in order to understand map more closely:



What happens in each case?

Now try to write the following functions:

• A function add that expects 2 arguments. It adds them and echos the result.

• A function multiply that also expects 2 arguments. It multiplies them and echos the result.

• A function operation that reads a global variable SEL and depending on its value calls add or multiply. It should pass all arguments supplied to operation further on to either add or multiply.

• A function calculate3 that takes a single argument and calls operation passing on this single argument and also the number “3” as the second argument to operation.

(optional) Write an encapsulating script that

• uses map to apply calculate3 all arguments on the commandline but the first.

• examines the first argument in order to set the variable SEL (e.g. the argument --add3 selects addition, the argument --multiply3 multiplication)

How much effort does it take to add a third option that allows to subtracts 3 from all input parameters

Exercise 6.7.

Make your script from exercise 6.6 sourceable and amend the

following script in order to get the functionality described in the comments:

