# COMP1562, Lab #5 Shell Programming II

|  |  |  |  |
| --- | --- | --- | --- |
| Lecturer | **Mariusz Pelc** | Phone | **020 83318588** |
| Office | **QM366** | e-mail address | **m.pelc@gre.ac.uk** |
| Office hours | **Mon 5-6pm, Thu 5-6pm** |  |  |

**(by M. Pelc, C. Ierotheou and K. McManus)**

**Description:**

**This exercise is mainly focused on more advanced aspects of Linux shell programming.**

**Learning Outcomes:**

**Students will be able to write more complex scripts in Linux shell. The report should contain results showing solutions for the tasks specified in the *Tasks* section**[[1]](#footnote-1)**.**

**TASKS**

### Typically, in time every Linux system has a number of symbolic links that are no longer tied with any existing file. This may happen due to various reasons, but ultimately this situation is something that system admin has to deal with. Very frequently the administrator decision is to delete such links. Your task is to write a script which would remove all symbolink links (symlinks) which are linked to no longer existing files / directories. The operation should be performed in the directory specified as the script execution parameter. NO READ COMMAND IN THE NON\_INTERACTIVE VERSION AND NO INFINITE LOOPS – IT WILL HANG SCRIPTCHECK!

### Assuming the script name is *scr1* and the directory you want to delete the symlinks is *Dir*, the script should be executed as:

### ./scr1 dir\_name

### Test your script before you upload it to the scriptcheck system in the following way:

### In any directory (you may want to create a special directory for the test purpose) create a number of files (contents is not important, these can even be empty files).

### Create separate symlinks for each of the files.

### Execute your script giving the directory name as the script argument (note: in this case none symlink should be deleted).

### Now delete one or two of the files (but not symlinks).

### Exectute again your script as previously (note: this time only the symlinks that point at the files you deleted should be removed).

### If the script did remove the proper symlinks, you can upload the script to the scriptcheck system.

### 

### Write a quiz-script which will provide some statistics about a quiz results (summarize how many good answers and how many wrong answers were provided). The script should use up to 3 execution parameters. When executed with two parameters (let’s call it interactive mode), it should assume, that the first file contains a list of questions and the second file contains correct answers to these questions. In such execution case your script should display questions one by one and wait for users response. The user’s response should be then compared with the correct answer to that specific questions read from the file containing answers and as a result the script should produce statistics as to how many good and how many wrong answers were provided.

### Questions should be formed so that either, YES or NO answer would apply. Questions file format can by any, whilst the answers format should contain only YES or NO answers, one answer per line. There should be exactly 10 questions in the quiz.

### Assuming that file containing questions is *qfile* and the file containing answers is *afile*, to run your script in the interactive mode it should be executed as:

### ./scriptname qfile afile

### How to test:

### Create a text file containing a number of questions.

### Create a text file containing correct answers to these questions.

### Execute your script:

### It should ask questions one at a time and then wait for your answer.

### Your answer should be then compared with the appropriate answer read from the file with answers,

### Depending on the correctness of your answer, the script it should count the answer and display statistics as:

### Number\_of\_correct\_answers Number\_of\_wrong\_answers

### However, if your script is executed with 3 command line parameters it should execute in automatic mode meaning that the first file should still be treated as questions, second as correct answers, but the third parameter should be treated as file containing user’s answers (for example sent on-line for verification). In such case the script should NOT display any questions, it only should compare user’s answers with the correct answers.

In the interactive mode the script should provide full statistics about the quiz results, hence the only thing your script is expected to **return** are **two numbers** without any additional prompt (like “The number of correct answers is: ” or “The number of wrong answers is: ”), just numbers, each number in a separate line, reflecting the number of correct and wrong answers. Neither, runtime messages nor error messages should appear on the screen (you may need to re-direct the standard error stream for operations that might be source of any kind of error as “*command 2>/dev/null”*).

In the automatic mode the script should only return **one number** corresponding to the number of correct answers. Neither, runtime messages nor error messages nor prompts whatsoever should appear on the screen (you may need to re-direct the standard error stream for operations that might be source of any kind of error as “*command 2>/dev/null”*).

### 

**Techniques/resources:**

**Solution of all the above tasks will require using any Linux machine (in the labs or at home).**

**Marking:**

**The solutions will be marking in the range 0-100%.**

**Deadline:**

**The solutions should be delivered within one weeks from the lab date.**

**☺ No marking sheet as this is tasks only week. Hence the scriptcheck system will do the whole marking.**

#!/bin/bash

right=0

wrong=0

question=10

i=0

if [ $# -eq 2 ]

then

for((i=0; i<question;)) {

i=$(expr $i + 1)

y=$i

echo "$(awk "NR==$y" $1)"

read -p "Enter Answer: " arg1

result=$(awk "NR==$y" $2)

if [ "$arg1" = "$result" ]

then

right=`expr $right + 1`

else

wrong=`expr $wrong + 1`

fi

}

echo "Number\_of\_correct\_answers " $right

echo "Number\_of\_wrong\_answers " $wrong

fi

if [ $# -eq 3 ]

then

for((i=0; i<question;)) {

i=$(expr $i + 1)

y=$i

ans="$(awk "NR==$y" $2)"

resultfile=$(awk "NR==$y" $3)

if [ "$resultfile" = "$ans" ]

then

right=`expr $right + 1`

else

wrong=`expr $wrong + 1`

fi

}

echo $right

fi

1. Not to mention commenting on the results / critical evaluation of the results ☺ [↑](#footnote-ref-1)