COMP1562 – Operating System
Laboratory 5

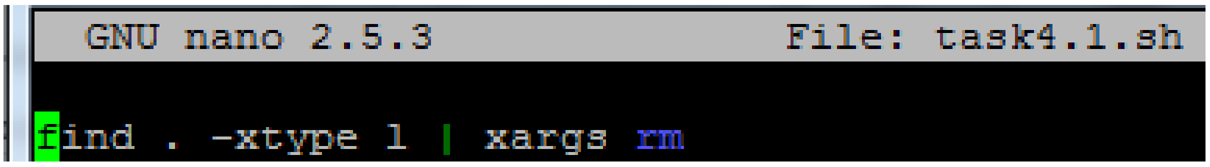
Shell Programming 2
Group ID: 21
Group Task: Task 4

Usman Basharat
Mohamed Aden
Yunus Hassan
Derek U. Mafohla

Table of Contents

Task 4.1 3
Task 4.2 4
Reflection 6

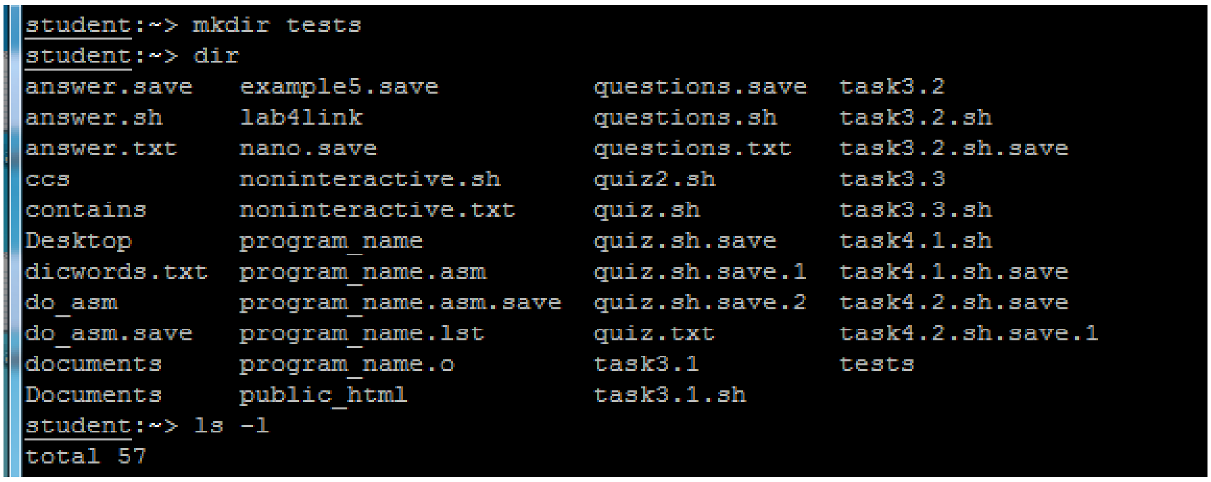
Task 4.1



```
  GNU nano 2.5.3                    File: task4.1.sh

find . -xtype l | xargs rm
```

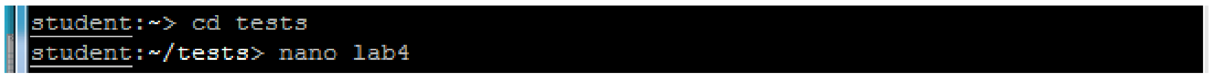
Figure 1 shows the code for Task 4.1

Figure 4.1 shows the code to execute broken symbolic links. As soon as the piping method is used to search for the broken links within the directory, it removes it. As I was doing this, I made a mistake of searching for symbolic links, and removing them. After reading the laboratory document, I realised it is the complete opposite. Therefore, when I ran the line of code through script check, it worked and I moved on to the next task. The screenshots below are the following of how this lab has been executed. The first movements were to create a directory and make an empty file. Once this was created, we need to create the link. Once the link was created, we make it broken by removing it and then we tested our code to remove it and it worked shown on Figure 6



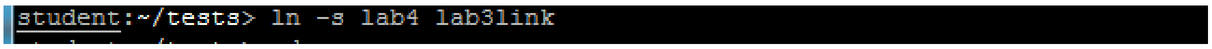
```
student:~> mkdir tests
student:~> dir
answer.save   example5.save         questions.save  task3.2
answer.sh     lab4link              questions.sh    task3.2.sh
answer.txt    nano.save             questions.txt   task3.2.sh.save
ccs           noninteractive.sh     quiz2.sh        task3.3
contains      noninteractive.txt    quiz.sh         task3.3.sh
Desktop       program_name          quiz.sh.save    task4.1.sh
dicwords.txt  program_name.asm      quiz.sh.save.1  task4.1.sh.save
do_asm        program_name.asm.save quiz.sh.save.2  task4.2.sh.save
do_asm.save   program_name.lst      quiz.txt        task4.2.sh.save.1
documents     program_name.o        task3.1         tests
Documents     public_html           task3.1.sh
student:~> ls -l
total 57
```

Figure 2 shows making a new directory



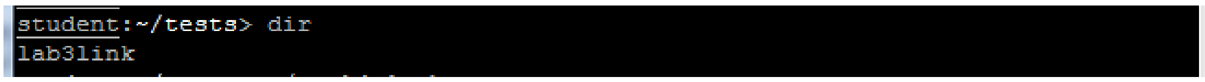
```
student:~> cd tests
student:~/tests> nano lab4
```

Figure 3 shows creating a new file



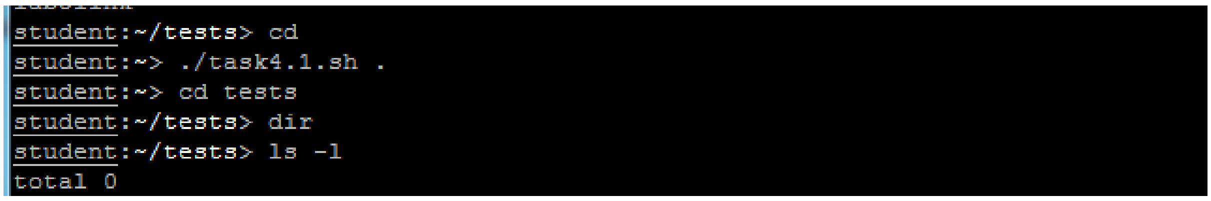
```
student:~/tests> ln -s lab4 lab3link
```

Figure 4 shows creating a symbolic link between the new file created



```
student:~/tests> dir
lab3link
```

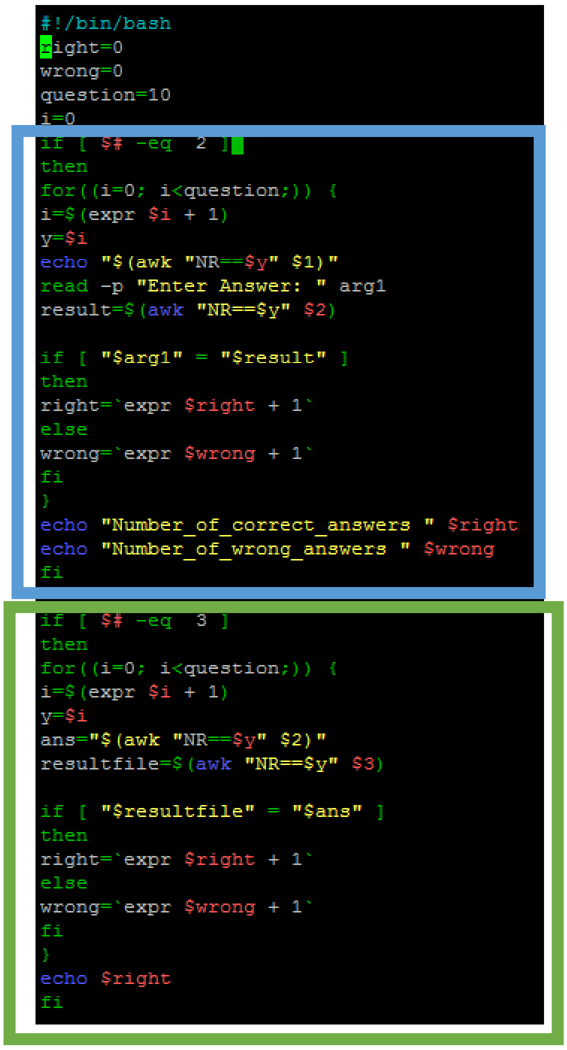
Figure 5 shows the broken link once the new file has been created



```
student:~/tests> cd
student:~> ./task4.1.sh .
student:~> cd tests
student:~/tests> dir
student:~/tests> ls -l
total 0
```

Figure 6 shows the symbolic link removed

Task 4.2



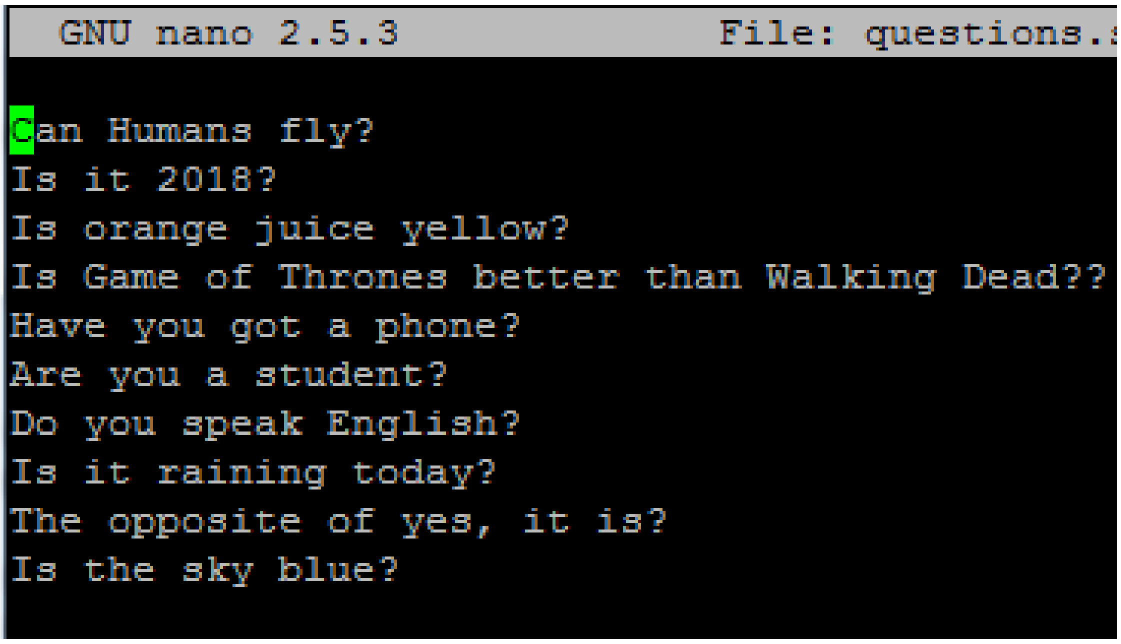
```
#!/bin/bash
right=0
wrong=0
question=10
i=0
if [ $# -eq  2 ]
then
for((i=0; i<question;)) {
i=$(expr $i + 1)
y=$i
echo "$(awk "NR==$y" $1)"
read -p "Enter Answer: " arg1
result=$(awk "NR==$y" $2)

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
```

The code is split between in two. This works by the first section highlighted in blue is recognised by the interactive mode of the quiz. This means that the user has to type in the answer they feel and they would receive the result. Once the user has typed in the result, this is matched by the correct answers that has been typed in. Each answer that is wrong, $wrong is added one and the same with $right.

However, the other section highlighted in green is the non-interactive part. This part shows that there are non-interactive answers that have been already typed in shown in Figure 9 and is compared to the actual answers shown in Figure 8. This only displays the correct answer. Once it executed, it matches the two and it displays how many answers are correct.

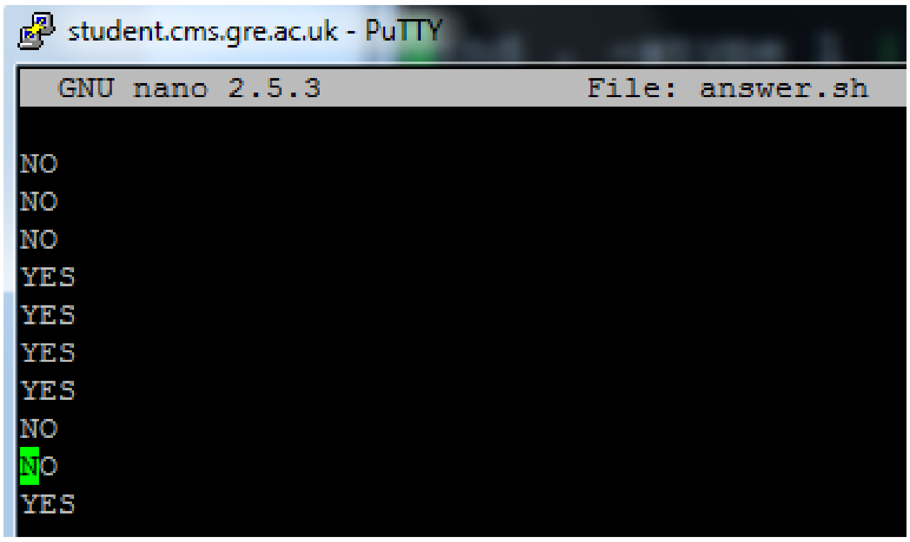
How the code is run for both is similar. The first loop runs 10 times, matching how many questions there are. Each time it runs, it uses the arguments to display the question and to match the answer. This is a clever loop, because once it is executed, it matches the arguments and selects which one is needed to execute.

Figure 10 and Figure 11 shows the difference between the two codes.



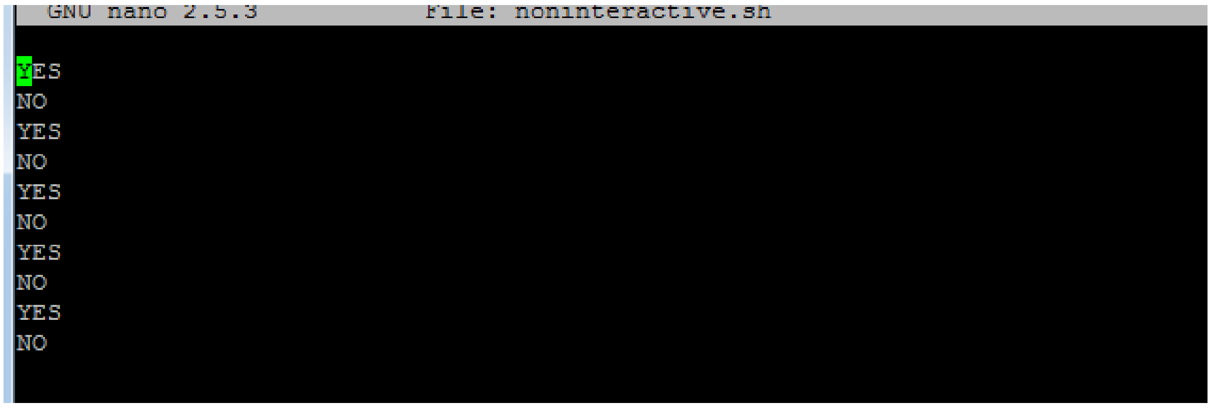
```
  GNU nano 2.5.3                 File: questions.s

Can Humans fly?
Is it 2018?
Is orange juice yellow?
Is Game of Thrones better than Walking Dead??
Have you got a phone?
Are you a student?
Do you speak English?
Is it raining today?
The opposite of yes, it is?
Is the sky blue?
```

Figure 7 shows the questions used for this task



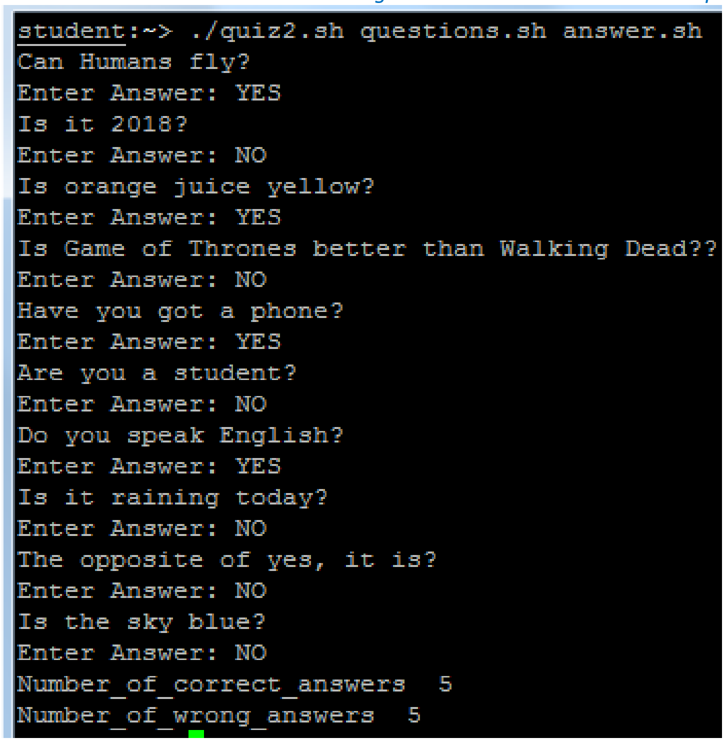
```
student.cms.gre.ac.uk - PuTTY
  GNU nano 2.5.3                  File: answer.sh

NO
NO
NO
YES
YES
YES
YES
NO
NO
YES
```

Figure 8 shows the answers to the questions for this task



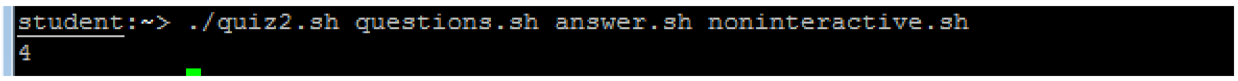
```
  GNU nano 2.5.3               File: noninteractive.sh

YES
NO
YES
NO
YES
NO
YES
NO
YES
NO
```

Figure 9 shows the non-interactive questions used.



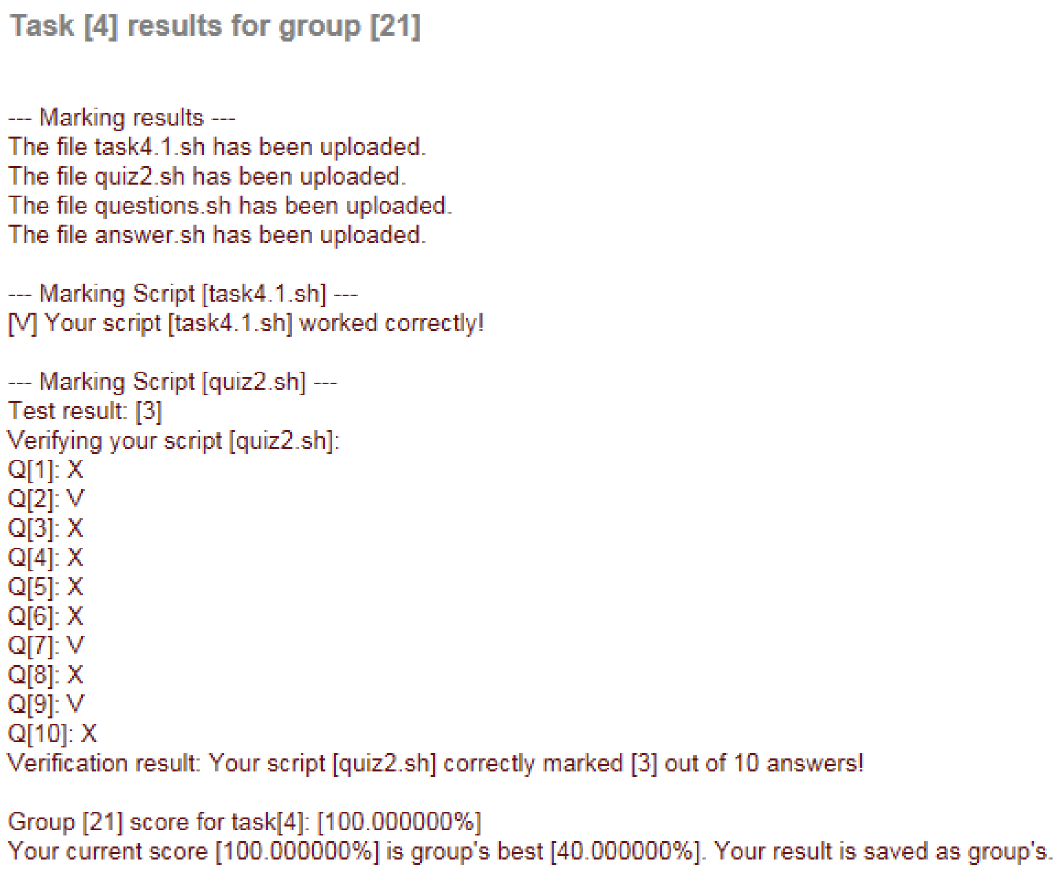
```
student:~> ./quiz2.sh questions.sh answer.sh
Can Humans fly?
Enter Answer: YES
Is it 2018?
Enter Answer: NO
Is orange juice yellow?
Enter Answer: YES
Is Game of Thrones better than Walking Dead??
Enter Answer: NO
Have you got a phone?
Enter Answer: YES
Are you a student?
Enter Answer: NO
Do you speak English?
Enter Answer: YES
Is it raining today?
Enter Answer: NO
The opposite of yes, it is?
Enter Answer: NO
Is the sky blue?
Enter Answer: NO
Number_of_correct_answers  5
Number_of_wrong_answers  5
```

Figure 10 shows the interactive quiz.



```
student:~> ./quiz2.sh questions.sh answer.sh noninteractive.sh
4
```

Figure 11 shows the non-interactive quiz



```
Task [4] results for group [21]

--- Marking results ---
The file task4.1.sh has been uploaded.
The file quiz2.sh has been uploaded.
The file questions.sh has been uploaded.
The file answer.sh has been uploaded.

--- Marking Script [task4.1.sh] ---
[V] Your script [task4.1.sh] worked correctly!

--- Marking Script [quiz2.sh] ---
Test result: [3]
Verifying your script [quiz2.sh]:
Q[1]: X
Q[2]: V
Q[3]: X
Q[4]: X
Q[5]: X
Q[6]: X
Q[7]: V
Q[8]: X
Q[9]: V
Q[10]: X
Verification result: Your script [quiz2.sh] correctly marked [3] out of 10 answers!

Group [21] score for task[4]: [100.000000%]
Your current score [100.000000%] is group's best [40.000000%]. Your result is saved as group's.
```

Reflection

During the time for this laboratory, I found this task tough again. I felt that the first one was not as tough as the second one, because we had to link three files together. It was hard to put the non-interactive part of the quiz in, but when I understood the first part of how I did it, it was easier to understand the non-interactive part. I felt that I spent much time on this the most, because it was the most challenging one out of them all. These challenges took time and effort to be completed and I felt that with my group members, we were able to complete this task and get the result above. Overall, I felt that I have understood the basic understanding and importance of how to run and produce the code effectively.