14.3.6 Practice Questions

Candidate: Ethan Bonavida (suborange)
Date: 12/8/2022 9:07:12 pm • **Time Spent:** 01:17

Score: 100% Passing Score: 80%

Question 1: ✔ Correct

Given the following bash script, what is the output if the user enters *Kali*?

```
#!/bin/bash
echo 'Which Linux distribution do you like? '
read distro

case $distro in
     ubuntu)
          echo "Ubuntu is based on Debian."
          ;;
     centos|rhel)
          echo "CentOS and RHEL are RPM based distributions."
          ;;
     windows)
          echo "That is not a Linux distribution."
          ;;
     *)
          echo "This is an unknown Linux Distribution."
          ;;
esac
```

- CentOS and RHEL are RPM-based distributions.
- Ubuntu is based on Debian.
- That is not a Linux distribution.
- ➡ This is an unknown Linux distribution.

Explanation

A case statement works well for testing two or more ways a condition could be evaluated. The case statement will check the input for a match. If no match is found, the catch all statement, represented by "*)", will be used. With user input of *Kali*, no match will be found, and the catch-all statement will be displayed.

References

- 14.3.1 Bash Scripting Logic
- 14.3.4 Branching
- 14.3.5 Bash Scripting Logic Facts

q_script_logic_lp5_case_statement.question.fex

Question 2: ✔ Correct

Anna, a technician, executed a command to display the contents of a file and received the output.

```
[user@linux ~]$ cat myfile.txt
at: myfile.txt: No such file or directory
```

Which of the following commands would Anna enter to find out the exit code that was returned by this command?

- exit
- env
- ➡ **echo $?**
- echo $1

Explanation

echo $? displays the exit code from the previously executed command. In this case, a value of 1 would displayed because the command failed. A 0 indicates no errors.

echo $1 does not display anything.

exit causes the shell to exit.

env displays the current environment variables.

References

- 14.1.4 Scripting Facts
- 14.3.1 Bash Scripting Logic
- 14.3.3 Exit Codes
- 14.3.5 Bash Scripting Logic Facts

q_script_logic_lp5_exit_codes.question.fex

Question 3: ✔ Correct

Given the following bash script,

```
#!/bin/bash
for i in $(ls)
do
  echo item: $i
done
```

Which of the following shows possible output if the script is executed from Bill's home directory?

- item: /home/sally
  item: /home/bill
  item: /home/mario
  item: /home/lucinda

- item: .bash_history
  item: .bash_logout
  item: .bash_profile
  item: bashrc

- ➡ item: Desktop
  item: Documents
  item: Downloads

- item: /
  item: /home
  item: /home/bill
  item: /home/bill/Documents

Explanation

The script will loop through the output of the **ls** command and display each item. In this case, the three folders Desktop, Documents, and Downloads were the only three items in Bill's home directory. The for loop iterated through the output.

The hidden file .bash_history, .bash_logout, and .bash_profile would not be include in the ls listing.

The /home/sally and other directories would not be included in the ls listing.

The root directory / and other directories would not be included in the ls listing.

References

14.3.1 Bash Scripting Logic

14.3.2 Looping

14.3.5 Bash Scripting Logic Facts

q_script_logic_lp5_for_loop.question.fex

Question 4: ✔ Correct

Given the following bash script:

```
#!/bin/bash
mynumber=5
guess=0
echo -e "I am thinking of a number from 1 to 10\n"
read -p "Enter guess: " guess
if (( guess == mynumber ))
then
 echo "That is correct!"
elif (( guess != mynumber )); then
 echo "Sorry, that is not my number!"
fi
```

Which of the following would be displayed if the number 12 is entered as the guess?

- error: number out of range
- 5
- That is correct!
- ➡ Sorry, that is not my number!

Explanation

Entering the guess of 12 will result in the output, "Sorry, that is not my number!" The if statement will evaluate 12 and compare it to 5. Since it is not equal, the next elif statement checks to make sure the number does not equal 5 and displays the message.

This bash script does not produce any of the other answers.

References

14.3.1 Bash Scripting Logic

14.3.4 Branching

14.3.5 Bash Scripting Logic Facts

q_script_logic_lp5_if_statement.question.fex

Question 5: ✔ Correct

You are writing a bash script that lists the contents of a file. You would like to have any stderr messages sent to a file.

Which of the following commands will write the error message to a file?

- **cat projects 2>&1 projects.err**
- **cat projects 1> projects.err**
- **cat projects > projects.err**
- ➡ **cat projects 2> projects.err**

Explanation

cat projects 2> projects.err redirects *stderr* to *projects.err*.

cat projects > projects.err redirects the output of the command to *projects.err*. It does not redirect *stderr* to the file.

cat projects 1> projects.err redirects the output to the file, not the stderr.

cat projects 2>&1 projects.err redirects *stderr* to *stdout* and displays any error on *stdout*. The file will not contain error messages.

References

- 14.1.4 Scripting Facts
- 14.3.1 Bash Scripting Logic
- 14.3.3 Exit Codes
- 14.3.5 Bash Scripting Logic Facts

q_script_logic_lp5_stderr.question.fex

Question 6: ✔ Correct

Which of the following statements is true about the command **myscript < mydata.txt**?

- The output of *mydata.txt* is stored in myscript, where it is processed.
- **myscript** outputs (stdout) data received from the mydata.txt input (stdin).
- The output of **myscript** is appended to *mydata.txt*.
- ➡ **myscript** receives input (stdin) from *mydata.txt*.

Explanation

myscript receives input (stdin) from mydata.txt.

References

- 14.1.4 Scripting Facts
- 14.3.1 Bash Scripting Logic
- 14.3.3 Exit Codes
- 14.3.5 Bash Scripting Logic Facts

q_script_logic_lp5_stdin.question.fex

Question 7: ✔ Correct

Given the command **ls > myfiles** which of the following describes the results?

- ➡ The stdout of the **ls** command is redirected to the myfiles file.
- The **ls** command outputs the contents of the myfiles file.
- The **ls** command takes the stdin from myfiles and displays the results.
- The **ls** command lists only the files that match those stored in the myfiles file.

Explanation

The stdout of the **ls** command is redirected to the myfiles file.

References

- **14.1.4 Scripting Facts**
- **14.3.1 Bash Scripting Logic**
- **14.3.3 Exit Codes**
- **14.3.5 Bash Scripting Logic Facts**

q_script_logic_lp5_stdout.question.fex

Question 8: ✔ Correct

Given the following bash script:

```
#!/bin/bash
declare -i count=5
until [ $count -lt 3 ]
do
  echo count $count
  count=count-1
done
```

Which of the following shows the output from this script?

- count 1
  count 2
  count 3
  count 4
  count 5

- count 5
  count 4

- ➡ count 5
  count 4
  count 3

- count 3
  count 4
  count 5

Explanation

This script produces the following output:

count 5

count 4

count 3

The until loop starts with the value of 5 as the count and continues to decrease the count by one until the number is less than 3. At that point, the until loop stops.

The script does not produce the other outputs.

References

14.3.1 Bash Scripting Logic

14.3.2 Looping

14.3.5 Bash Scripting Logic Facts

q_script_logic_lp5_until_loop.question.fex

Question 9: ✔ Correct

Given the following command sequence:

```
echo 'blue orange green brown' | while read a b c d; do echo output: $b $c $a $d; done
```

Which of the following is the correct output?

- output: blue orange green brown
- ➡ output: orange green blue brown
- output: b l u e
- output: b c a d

Explanation

The results of the while loop will produce *output: orange green blue brown.* The while loop will read in the four values from the **echo** command and display them in a different order based on the variables $b $c $a $d.

output: blue orange green brown is the incorrect result since the second echo displays the input in different order.

output: b l u e is incorrect because the *read* command will read an entire word delimited by spaces into the variables.

output: b c a d is incorrect because $b $c $a $d are variables and contain the values read from the first **echo** command.

References

14.3.1 Bash Scripting Logic

14.3.2 Looping

14.3.5 Bash Scripting Logic Facts

q_script_logic_lp5_while_loop.question.fex