Graded Quiz: Test your Project understanding
Graded Quiz • 30 min

Due Jul 6, 12:29 PM IST

! **Try again once you are ready**

TO PASS 70% or higher

Try again

GRADE
61.15%

Graded Quiz: Test your Project understanding

LATEST SUBMISSION GRADE

61.15%

1. What does the acronym Bash stands for?

0 / 1 point

○ ~~Basic Animated SHell~~

○ Bourne-Again SHell

○ Backus-Naur Script

● Bold Action Shell

! **Incorrect**

Incorrect. Please review 'Task 1' for more information.

2. Which of the following are **command languages**? Select all that apply.

0.75 / 1 point

☑ lodash

! **This should not be selected**

Incorrect. Lodash is a JavaScript utility library. Feel free to claim the name for your next command language!

☑ bash

✓ **Correct**

Correct! Bash - Bourne Again SHell, was developed by Brian Fox and first released in 1989. It is the default shell on most Unix-like operating systems today.

☑ sh

✓ **Correct**

Correct! The Bourne shell *'sh'* was developed by Stephen Bourne. It was released in 1979.

□ ~~smash~~

3. Which of the following commands will display the text *"I am great! :)"* when executed through the command line? Select all that apply.

1.2 / 2 points

☑
```
echo I am great! :)
```

! **This should not be selected**

Incorrect. Executing the command will throw an error. Please watch 'Task 2' where escaping special characters is discussed.

☑
```
echo "I am great! :)'
```

! **This should not be selected**

Incorrect. Executing the command will throw an error. Please watch 'Task 2' where escaping special characters is discussed.

☑

```
echo I am great! :\)
```

✓ **Correct**

Correct! The backslash "\" can be used to escape special characters such as circle brackets ")".

☑

```
echo "I am great! :)"
```

✓ **Correct**

Correct! Surrounding the expression with double quotes allows you print special characters such as circle brackets ")".

☑

```
echo 'I am great! :)'
```

✓ **Correct**

Correct! Surrounding the expression with single quotes allows you print special characters such as circle brackets ")".

4. What will be the current directory after executing the *cd* command without any arguments?

0 / 1 point

○ The *root* directory.

○ The *home* directory.

● The current directory will remain the same.

○ ~~The parent directory of the current directory.~~

! **Incorrect**

Incorrect. Please refer to 'Task 3' for more information.

5. What will happen after executing the following command if a directory with the name '*my-new-directory*' **already exists** in the current directory?

1 / 1 point

```
mkdir my-new-directory
```

● Error will be displayed.

○ New empty directory with the name *my-new-directory (1)* will be created.

○ The system might crash.

✓ **Correct**

Correct! The *mkdir* command throws an error if the specified directory already exists.

6. Which command should I use if I want to **append** the content *"I am new here"* to the file *foo* without overwriting the file?

1 / 1 point

○

```
echo "I am new here" | foo
```

○

```
echo "I am new here" foo
```

●

```
echo "I am new here" >> foo
```

○

```
echo "I am new here" > foo
```

✓ **Correct**

Correct! Using the greater-than sign twice - ">>", will append the text to the file instead of overwriting it.

7. What does the *ls* command do?

2 / 2 points

● lists directory contents

○ changes the current directory

○ displays *ls* on the screen

○ creates a new file

✓ **Correct**

Correct! If you want to learn more about the available options execute "*man ls*".

8. Executing which command will display the **number of files** in the current directory?

1 / 1 point

●
```
ls | wc -l
```

○
```
ls -l
```

○
```
ls > wc -l
```

○
```
ls wc -l
```

✓ **Correct**

Correct! Piping the output of the *ls* command to the *wc -l* command will display the number of files in the current directory.

9. Executing which of the commands will rename the file *photo1.jpg* to *photo.jpg*?

0 / 1 point

[redacted]

●
```
rn photo1.jpg photo.jpg
```

[redacted]

○
```
mv photo.jpg photo1.jpg
```

! **Incorrect**

Incorrect. Please refer to 'Task 7' where renaming files is discussed.

10. What command can you execute to remove a directory with the name *foo*?

0 / 1 point

rm -r foo

! **Incorrect**

Incorrect. Please refer to 'Task 7' where removing files and directories is discussed.

11. What's the difference between terminal emulator and Bash?

1 / 1 point

10

+50
Bash is one of the popular command-line shells, programs whose chief job is to start other programs (in addition to some auxiliary functions).
The command-line part means you control it by typing commands one line at a time.
Properly speaking, a GUI you use to start programs by double-clicking on icons is also a shell, but in practice by "shell" people mostly mean command-line ones.

All modern command-line shells take their input and send their output as abstract streams of characters, and the other ends of those streams can be connected to a keyboard, a printer, a file, another program. The shell mostly doesn't care - it reads the characters, interprets them as commands telling it to run other programs, and writes back characters such as "command not found". When it runs another program, by default it connects the inputs and outputs of that program to the same streams.

Now, Terminal is a program that provides a graphical interface between the shell and the user. It receives from the shell e.g. the characters "command not found" and figures out how to display them to you - with what font, where on the screen, in what colour, whether there should be a scrollbar. When you press some keys, it figures out whether to send them on to the shell as characters (e.g. ls -l), or to interpret them on its own (e.g. ⌘C).

When you open the Terminal app, it automatically opens a shell to connect you to. In its settings, you could choose a different shell from Bash. If you're feeling cheeky, you could even make it use a program that isn't a shell at all - not too useful, but it demonstrates how Terminal cares only about passing characters in and out, not about what the shell does with them.

What happens when you type bash into Bash (through Terminal)? It starts the program Bash - that is, another copy of itself inside itself.

✓ **Correct**
The terms Bash and terminal emulator are discussed in 'Task 1'.