**Build a Build Script**

One common use of bash scripts is for releasing a “build” of your source code. Sometimes your private source code may contain developer resources or private information that you don’t want to release in the published version.

In this project, you’ll create a release script to copy certain files from a **source** directory into a **build** directory.

**Tasks**

**0/13Complete**

Mark the tasks as complete by checking them off

**Start the script**

**1.**

Take a look at the **build** and **source** folders. The objective of our script is to copy files from **source** to **build**, with a couple of exceptions and modifications.

Get started on the script by adding a header to **script.sh**, identifying the type of script.

Hint

Start the script with:

#!/bin/bash

**2.**

Let’s welcome the user to the script. Feel free to use your own style here.

Make sure to save your script. Test your script in the terminal using ./script.sh.

Hint

For example:

echo "🔥🔥🔥Beginning build!! 🔥🔥🔥"

**3.**

Since we are creating a new build, let’s verify with the user that they have updated **changelog.md** with the current release version.

The first line of the file contains a version number with markdown formatting like so:

## 1.1.1

Read the first line of this file into a variable firstline. You can use the linux command [head](http://www.linfo.org/head.html) for this purpose.

Hint

By default, head returns 10 lines. But in this example, we want the first line. We can use the -n argument to specify the number of lines:

firstline=$(head -n 1 source/changelog.md)

**4.**

We want just the version number without the markdown formatting. The command [read](http://linuxcommand.org/lc3_man_pages/readh.html) can be used to split a string into an array using the -a argument.

Split the string firstline into the array splitfirstline.

The syntax for splitting a string foo into an array bar is:

read -a bar <<< $foo

Hint

To split firstline into an array and assign it to splitfirstline:

read -a splitfirstline <<< $firstline

**5.**

Now we are ready to set the value of the version of the script. It is located in index 1 of the array splitfirstline.

The syntax for accessing the value at index of an array foo is:

${foo[index]}

Save the version to a variable, version.

Print a statement to the terminal notifying the user of the version they are building.

Hint

The version is at index 1, so the following code will get the version number and print it to the terminal:

version=${splitfirstline[1]}

echo "You are building version" $version

**User Input**

**6.**

Let’s give the user a chance to exit the script if they need to make a change to the version.

Ask the user to enter “1” (for yes) to continue and “0” (for no) to exit.

Assign their response to the variable versioncontinue.

Hint

The following will get user input and assign it to versioncontinue:

echo 'Do you want to continue? (enter "1" for yes, "0" for no)'

read versioncontinue

**7.**

Add a conditional. If the user said “1” to the continue question, we will execute the rest of our script. For now, respond “OK”.

If the user did not, tell them “Please come back when you are ready”.

Hint

Here we use -eq because we are checking for equality between two numbers:

if [ $versioncontinue -eq 1 ]

then

echo "OK"

else

echo "Please come back when you are ready"

fi

**Copying files**

**8.**

Now we want to copy every file from **source** to **build** except for **secretinfo.md**.

Within the positive conditional (where we told the user “OK”), start by iterating over all the files in the **source** directory and printing their names to the terminal.

Hint

Use the wildcard \* to access files within the source directory:

for filename in source/\*

do

echo $filename

done

**9.**

Check if the filename is “source/secretinfo.md”. If it is, inform the user that it is not being copied.

Otherwise, inform the user that it is being copied.

Make sure to use spaces in your string conditional.

Hint

Here we are comparing two strings, so we use == to check equality:

if [ "$filename" == "source/secretinfo.md" ]

then

echo "Not copying" $filename

else

echo "Copying" $filename

fi

**10.**

Now we can actually copy the files. After informing the user the file is being copied, copy the file into the build directory.

You can use the terminal to make sure the right files have been copied:

ls build/

Hint

We are copying the file using the cp command:

if [ "$filename" == "source/secretinfo.md" ]

then

echo "Not copying" $filename

else

echo "Copying" $filename

cp $filename build/.

fi

**Listing files**

**11.**

The final thing we want to do is list the files in the **build** directory for the user.

Outside of the loop over the filenames in the directory, use the script to change the directory to the build directory. So that we don’t forget, also add the command to change back to the directory with the script.

Hint

Use cd to change directories:

cd build/

# We'll add more code here later

cd ..

**12.**

Add code to notify the user what files are currently in the build directory.

Be sure to reference the version in your message.

Hint

Use ls to list the contents of a directory:

echo "Build version $version contains:"

ls

**13.**

You now have a build script for this repository. Feel free to play around with making it more robust. Some ideas:

* Copy **secretinfo.md** but replace “42” with “XX”.
* Zip the resulting **build** directory.
* Give the script more character with emojis.
* If you are familiar with git, commit the changes in the build directory.