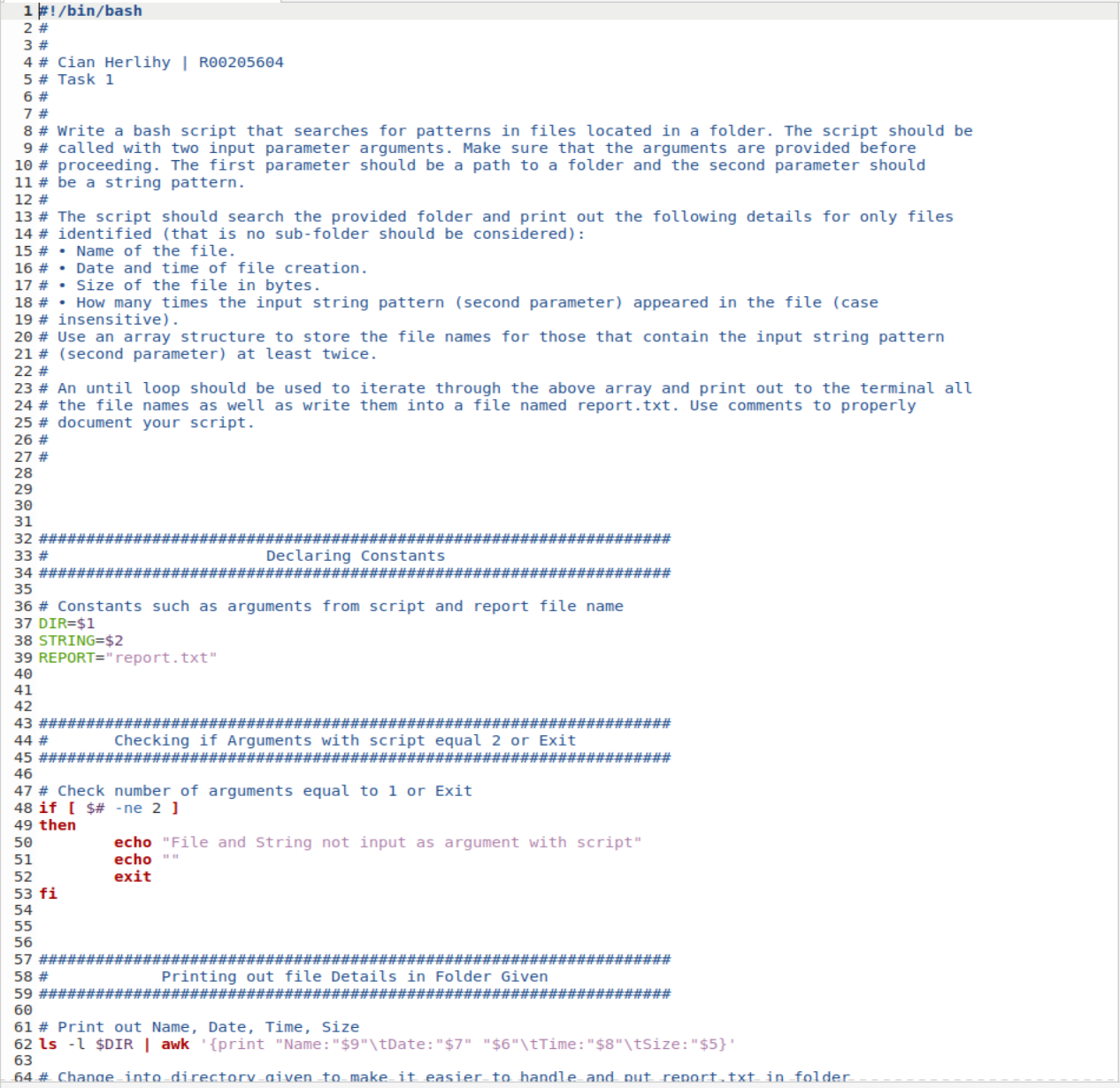
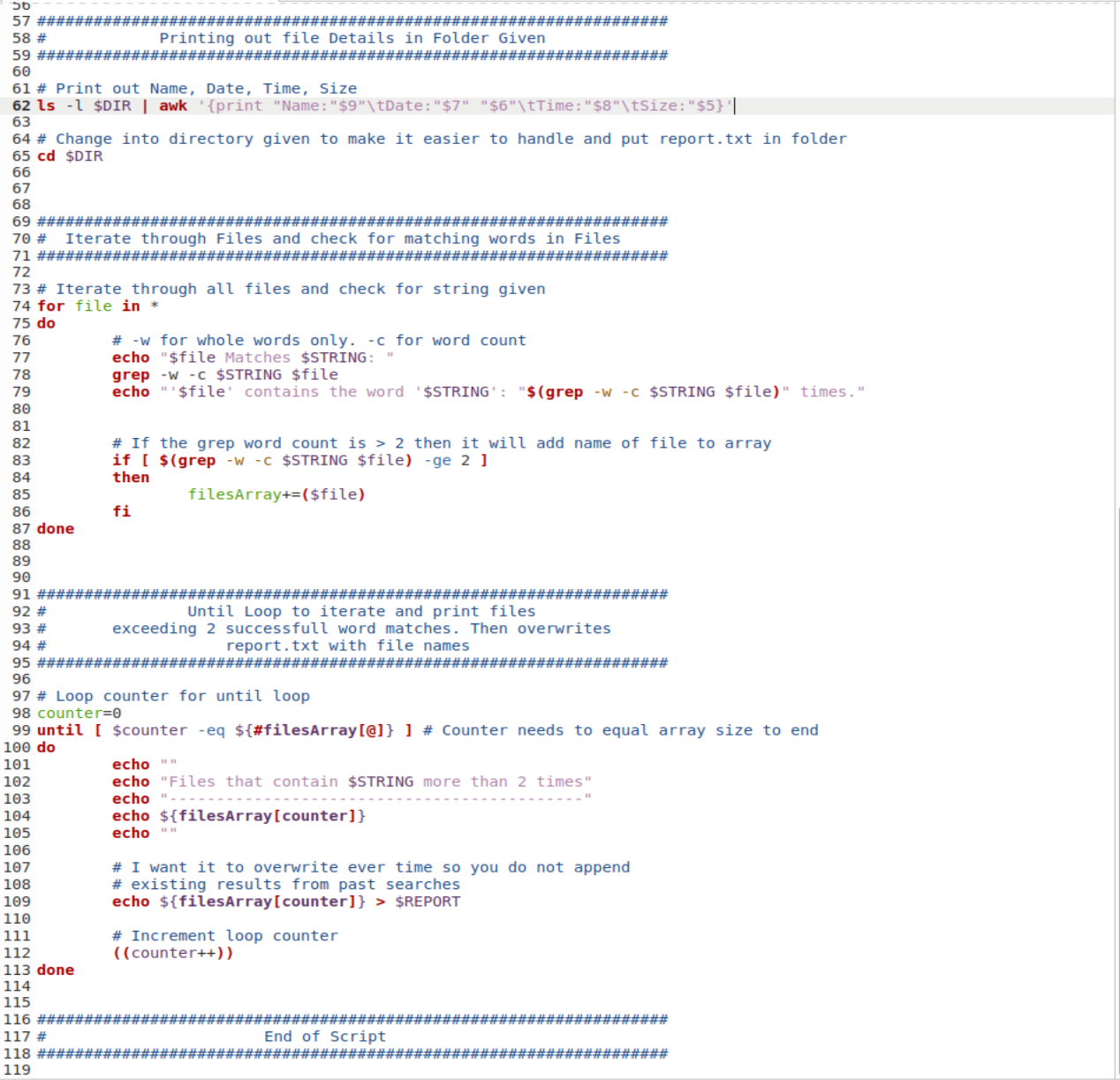
Assignment 1 – System Scripting

Cian Herlihy | R00205604

# Task 1





For task 1, I needed to take in 2 arguments, so I made sure to check if only 2 arguments were given. If it was not exactly 2 then it will give an error. I declared constants for the arguments because it gives the script more understanding than just seeing ‘$1’ and ‘$2’.

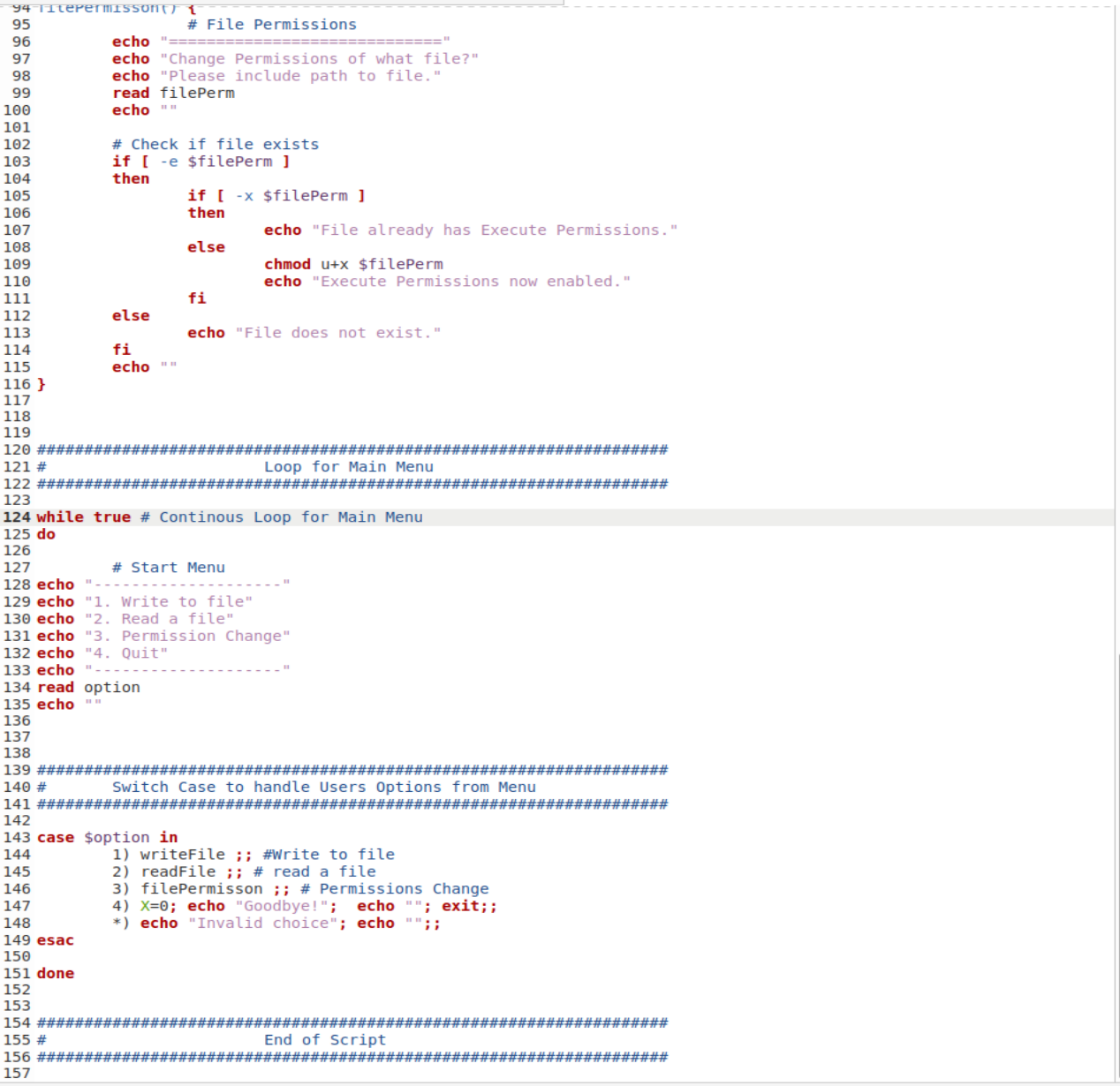
I iterate through the files using a for loop and in this for loop I check within the files using grep if the file contains any matching patterns to the string and I made sure to add -w next to -c to match the whole word. For example, I caught 6 matches in my file with the word test when there was only 5. This was because it counted the word ‘testing’ as a match for ‘test’. I did not want that outcome, so I fixed that error. On lines 77 -79 could be excluded since you do not need to print out each files word count but for testing purposes, I had it printing but simply commenting this out would work just as good.

I then have gathered all the files that contain more than or equal to 2 matching word counts to an array and I iterate through the array to then print off what files met this requirement. I then redirected the output to a report.txt file but I purposefully left it as overwrite so I did not need to clear it every time and get mixed up results with past running of the script.

# Task 2







The task 2 script should allow the user to write to a file, read a file and change the permissions to allow for a user to have execute permissions. I accomplished this by using a switch case for my menu. I echoed out the menu and then let the user select using numbers 1-4. I feel like this is a very simple way.

I then use a switch case statement to call on functions to do the work that was intended for that option. I started with option 1 being “Write to a File” which then calls on the function write File. This function is located above the switch case statements because bash would not have seen the function if it was below it causing an error.

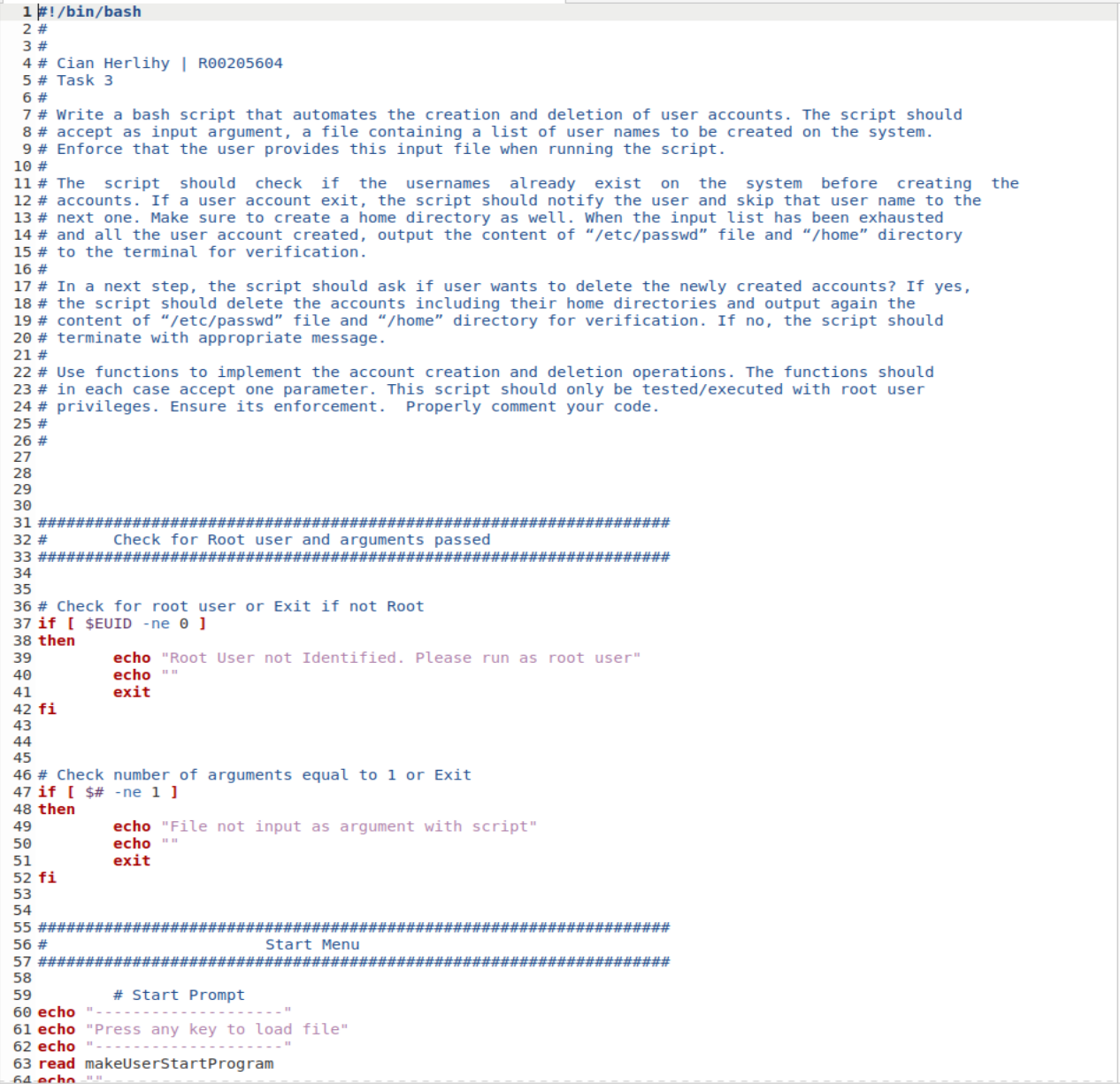
In this function I append content to the file if it already exists or it will create the file if it does not exist. The user can keep typing while skipping some lines too. The way to stop the script from looping when you are done inserting information into a file is to type ‘stop’ on its own line. It will not recognise the word in the middle of the line allowing you to be unrestricted from using that word.

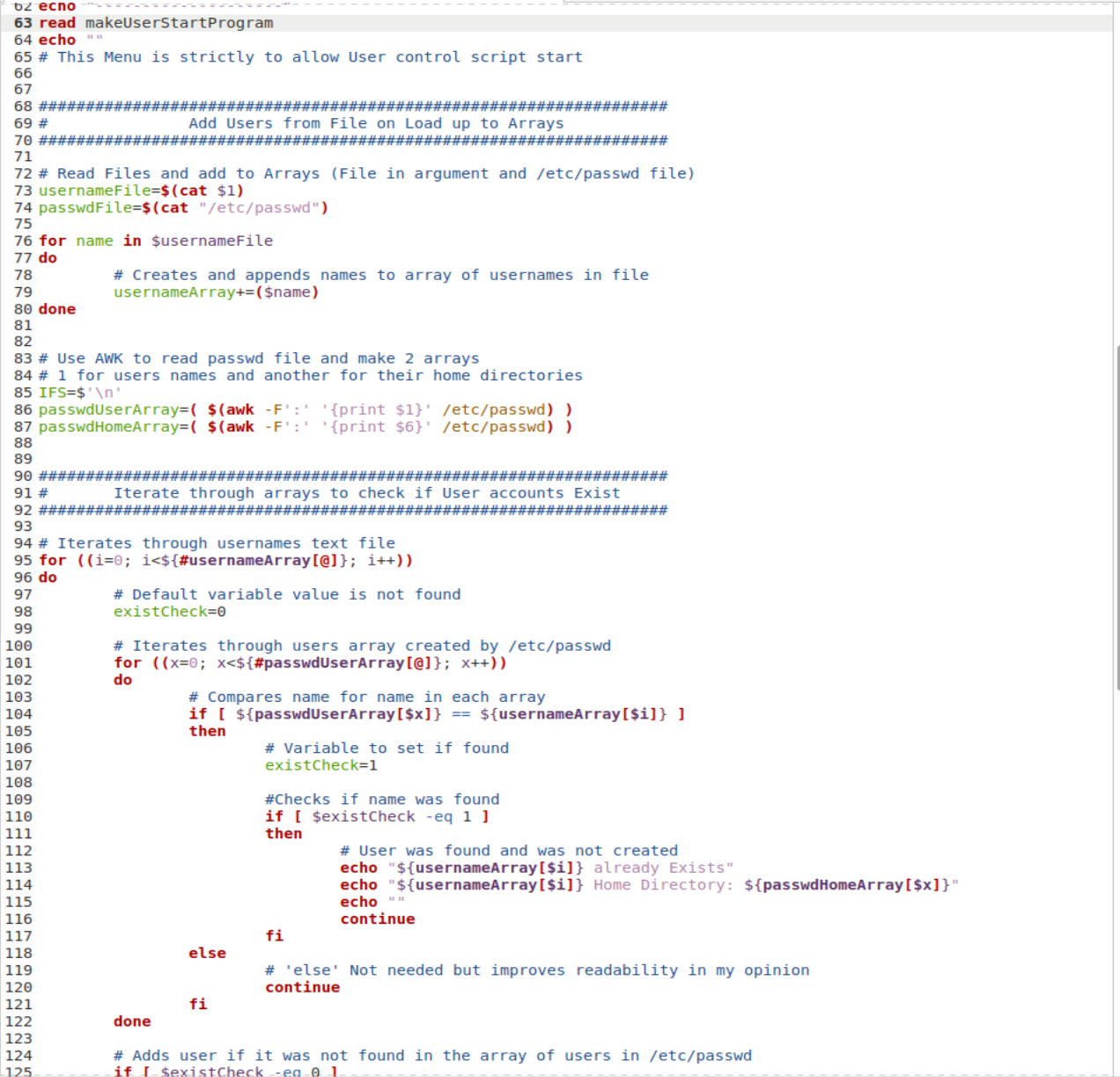
The second function then reads a file and firstly it checks if it exists. If it doesn’t exist, then it will try read the file. If it is empty, it will prompt the user that it is empty. Otherwise, it will read the file to the terminal.

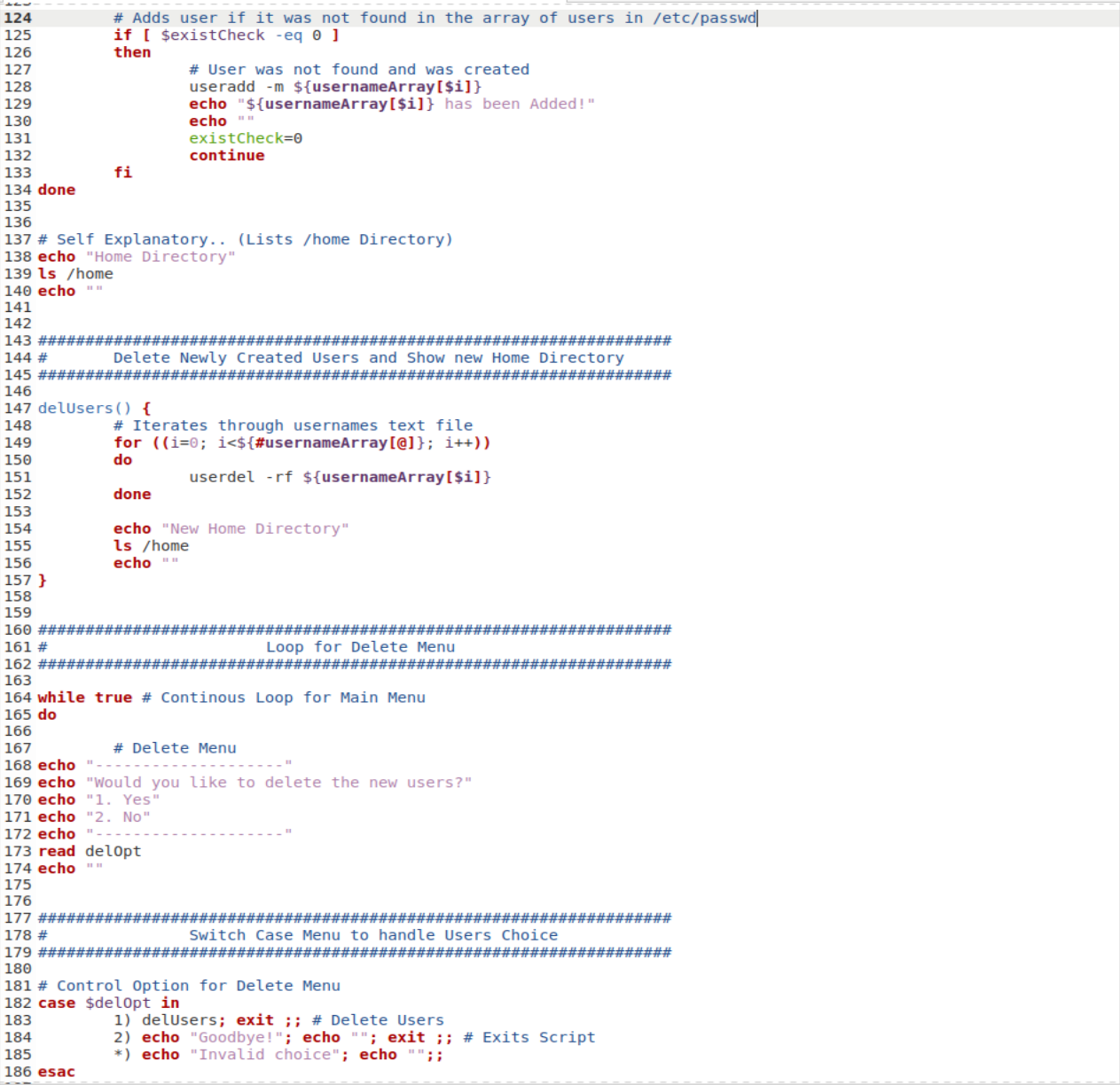
3rd function then changes the permissions of a given file to allow execute permissions for the user. If the file already has execute permissions, it will prompt the user that it already has the permissions.

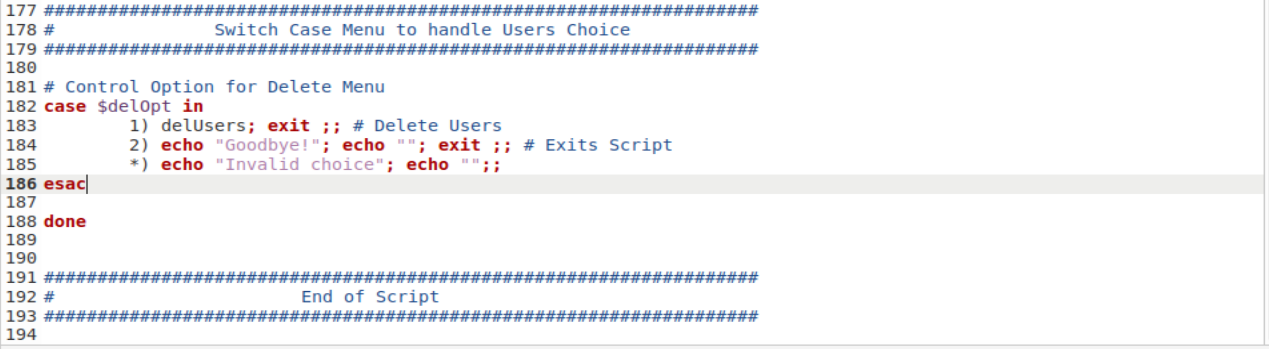
Lastly the program exits the infinite menu loop by choosing 4 as their option. If they choose an option that doesn’t exist like ‘5’ then it will prompt the user that it is not a valid option and then loop it again for a better choice.

# Task 3









For Task 3 I start off by checking if the user is root and then if it has 1 argument passed and only 1. Once it meets these requirements it can then move on to the actual functionality of the code. I auto load up the file given as an argument and create the users with the names in the file. I make the user press a key to proceed with the loading to give the user more control. They could exit the program by pressing ‘CTRL’ + ‘C’ and this would exit the script abruptly. When the user presses a button, it will proceed to load and create all the users. It will first check if there’s any users with that name already existing and if so, then do not try and create another with the same name.

If it has added the user or found a duplicate, then it will notify the user. The user will then be prompted with the delete menu. This give them the option of deleting the users that were just added to the system all while displaying the /home folder to prove the users were added correctly. If the user selects yes, then it will delete the users off the system and exit the script. If they want to keep the users, then they select no, and it exits for them. I have a switch case statement in a while loop that filters out invalid answers/selections from the menu.