ULI705 Assign# 1 – Bash Shell Scripting

Worth 10%

Part 1: Worth 4%, due Sat, Oct 14, by midnight
Part 2: Worth 6%, due Tues, Oct 24, by midnight

Part 1: File Permissions **Worth 4%** **Due: Sat, Oct 14, by midnight**

Write a Bash shell script (~/assign1/perm.sh) to do the following:

Prompt the user for a filename. Check the filename the user supplies to you and make sure that it exists. If not, give the appropriate error message to the user. Loop upto 3 times if invalid filename is given which does not exist. If valid filename which exists, is still not received from user, then exit the script without further processing.

If the file user indicated exists, then execute the command that displays the long list of information about the file, including its permissions. Now ask the user what permissions they would like this file to have. User should enter the required permissions in the format : r w x r - x - - -
Now take the permissions that the user has entered, make sure they are in the correct format ie. It can only have r, w, x or a hyphen in the permissions and nothing else. Also ensure that the r, w and x are in the correct positions. If not, give the appropriate error message to the user. Loop upto 3 times if invalid permissions are given. If valid permissions are still not received from user, then exit the script without further processing.

If the user's request for permission was valid, then convert those permissions into the corresponding octal number. For example, if the permissions for the file were r w x - - - - - - then it should convert to 700 as the corresponding octal number. Display an appropriate message to the user telling them the full 'chmod' command that they need to type, in order to change the file permissions to what they have asked for.

Validation
For each of the following, if blank or invalid data entered, loop and prompt till valid input received. After 3 tries, if data is still invalid, then stop script with appropriate message

- # of parameters entered
- Filename – filename that user supplies should exist
- Permissions – should be r, w or x only and must be in the right format and the right order
- Convert permissions to appropriate octal number and display the appropriate command that the user should type in order to change current permissions to what they want.
- Blank data, spaces or invalid data should not be allowed
- Script should not abort or crash at any point

Other requirements:

- All prompts to the user should be meaningful and give enough information to the user to enable them to provide required input
- Meaningful error messages should be given to user whenever error occurs
- All variable names should be meaningful and appropriate
- Appropriate exit status (0 or non-zero) should be set when exiting a condition
- Document your code throughout to describe what is happening
- Your code should use 'functions', be well organized and easy to follow
- Script should be located in ***~/assign1*** directory on zenit server and should be named ***perm.sh***

Part 2: Student Marks Report **Worth 6%** **Due: Tue, Oct 24 by midnight**

Write a Bash Shell Script (~/assign1/marks.sh) to do the following:

Get input from the user regarding student information and marks, then validate and display the results. First prompt the user to get student information (first name, last name and student number). Ensure that correct number of parameters are entered and that data is valid (see validation rules below). If not, keep looping upto 3 times to get correct number of parameters containing valid data. If you still don't get valid data then stop the script.

Once you have the name and number for a student, prompt to get 3 course-codes with corresponding marks for each course. Again, loop upto 3 times till you obtain valid data, if not, then stop the script.

The script should check the marks to see in which course you got the highest marks, and display that result in a meaningful manner. If the highest mark was obtained in more than one courses, you need to display message showing all courses that have that highest mark.

Validation

For each of the following, if blank or invalid data is entered, loop and prompt till valid input is received. After 3 tries, if data is still invalid, then stop script with appropriate message

- # of parameters entered
- Student name – Should have first name and last name, each of which should ensure first letter is uppercase and the rest is lowercase (to keep it simple, assume there is no middle names)
- Marks – numbers 0 to 100 only. Marks can be integers (eg. 90) or could have one decimal place (eg. 90.5) If user enters any leading zeros, then they should be stripped off (eg. 090.5 should become 90.5).
- Course code – 6 characters (first 3 uppercase alpha, next 3 numbers eg. ULI705)
- Student number – 6 digit positive integers only, including any leading zeros. Eg. 006574 is valid. Do not strip the leading zeros from the student number.
- Blank data, spaces or invalid data should not be allowed
- Script should not abort or crash at any point

Other requirements:

- All prompts to the user should be meaningful and give enough information to the user to enable them to provide required input
- Meaningful error messages should be given to user whenever error occurs
- All variable names should be meaningful and appropriate
- Appropriate exit status (0 or non-zero) should be set when exiting a condition
- Document your code throughout to describe what is happening
- Your code should use 'functions', be well organized and easy to follow
- Script should be located in ***~/assign1*** directory on zenit server and should be named ***marks.sh***

<u>To submit the assignment:</u>

1. Please do NOT email me anything when you are done. All final working code should exist in both partner's directories. I could go into either of the 2 partner's zenit accounts to test your scripts.
2. If date/time stamp on any of your files is later than the due date/time, then 'late penalty' of 1% grade for each late day will be applied.
3. Make sure you test with the valid/invalid data and valid/invalid parameters
4. I could ask you to explain any part of your code to me, so please make sure you understand it