Bash Grader Project Report

Ramachandran S

April, 2024

Contents

1 Basic Logic of the Script

The bash script, named `submission.sh`, is designed to manage and interpret CSV files containing student data various for exams. It provides a group of commands to perform various operations on these files and make our lives easier.

2 Utilities Used

The script primarily utilizes basic Bash scripting, which includes `if-else, for-while, arrays` for performing the basic functionalities along with `sed` and `awk`. Additionally, The script also utilises various flags in-built in bash for making conditional statements better. For customizations and advanced functionalities, I have used Python to make functioning easier and display better graphs and statistics. Some of the important Flags and Commands used in Bash are as follows:

- **-n <string>:** Returns True if the string is non-empty. It was used to check if a particular line is present in a file by using grep and checking if the output of grep is empty.
- **basename <string> <string-to-remove>:** Removes the <string-to-be-removed> from the <string>
- **openssl rand -hex 8:** OpenSSL rand is a command that generates random data `-hex 8` specifies it should be in hexadecimal format and of length 8 bytes.
- **read -p ”prompt message” variable:** Displays the prompt message before reading input and stores it in the variable.

Following are the Python Libraries used for better Customization:

- **Pandas:** To make reading data from csv files easier using the inbuilt functions.
- **Numpy:** Using it's inbuilt functions, evaluating statistics was made easier.
- **Matplotlib:** To plot graphs of the marks scored by students and to plot stats.
- **Sys:** To read command-line arguments written while executing a Python file
- **Scipy:** To find the percentile of a student and display it when the Show function is called
- **Tabulate:** To Tabulate exams and corresponding marks in a neat way and display it in terminal.

3 Pre-Requisites

This Bash Script uses a couple of Linux Terminal Commands and a few Python modules to make the code more efficient. So, it is better to make sure the host pc has all the following to make sure the script runs without throwing compilation error.

- **Sed** and **Awk**: The bash script uses sed and awk commands. So it is better to check for their installations. To check for their installations, use:

  ```
  sed --version
  awk --version
  ```

- **Python3:** The Bash script uses python3 to execute the python files that are made for customization. So it is better to check if python3 is installed in your pc or not. To check for it's installation, use:

  ```
  python3 --version
  ```

- **bc** command: bc command is used in the bash script for playing around with floating point numbers and arithmetic operations related to them. But it might not be available in all systems. In order to install them, use:

  Windows/Ubuntu based systems : `sudo apt install bc`
  macOS : `brew install bc`

- **Python Libraries:** Various Python Libraries are used in the bash script. If they are not installed in the host system, then it might result in an error. To install any library, use the following command:

  ```
  pip install <library_name>
  ```

  List of Libraries used are:

  - Numpy
  - Matplotlib
  - Pandas
  - Sys
  - CSV
  - Tabulate
  - Scipy

4 Basic Functionalities

CSV Processing Functions:

- **combine**: Reads all CSV Files and compiles the marks into `main.csv`. It also creates `Roll_Numbers.txt` file that stores Roll Numbers and Names seperated by a space. One Assumption is that Name corresponding to a particular Roll Number is same throughout every file.
- **upload**: Copies a new CSV file into the script's directory for future processing and personalized error message displayed if the file doesn't exist.
- **total**: Adds a new column in `main.csv` containing the total marks of each student.
- **update**: Updates marks of a student in `main.csv` and individual files.

Git Functions:

- **git_init**: Initializes a remote directory to store versions of the script and files.
- **git_commit**: Copies the current version of files to the remote directory and logs the commit.
- **git_checkout**: Reverts the current directory to a specified commit.

5 Customizations

The script has been included with several customizations to enhance the functionality and usability of our script:

- **Stats:** Added statistics commands to calculate mean, median, standard deviation, min, max and qth-percentile of student marks. An Added feature is that you can calculate it for either the total marks or for any individual exam.
- **Display:** Implemented graph generation using Python modules to visualize student performance. The two graphs available are:
  1. **Histogram:** Shows frequency of student marks split into small ranges.
  2. **Stats:** displays a bar graph that has all stats in it. An added feature is that you can display any individual's mark in the same graph so that you can compare where you stand with respect to the stats.
- **Show:** Creates a mini Report Card sort of thing and displays it in terminal. It prints the Roll Number and Name of the student. After that, the marks scored in individual exams and the total scores are printed in a tabular manner. Then, the percentile of the student's total score and their attendance in the exams is displayed.

- **List:** Lists all the functions along with the way they are supposed to be used.
- Extended Git functionality by adding more commands and optimizing existing ones.
  1. **git_add:** Adds the file to the staging area
  2. **git_log:** Displays commit id and commit message of all the previous commits.
  3. **git_status:** Prints the changes done to the working directory after the latest commit
  4. **git_clone:** Similar to the actual git clone, it clones the repo into a directory you specify.
  5. **git_amend:** Modifies the latest commit, similar to git amend.
  6. We do know that Git becomes better when you can create many repositories and manage set of files in different repos to make working better. For this, I have made sure that every time you want to create a new repo in any folder, you can use git_init in the new directory. The git functions you use after git_init gets added to the latest repository that was initialised. Now to make sure you can access other repos as well, I have added two functions so that user can see which repo he is currently accessing and switch between repos.
     - **Switchrepo:** Switches between Repository
     - **Currrepo:** Displays the current repository that the user is accessing.

6 Usage of Functions

The following section contains detailed way to use every function:

- **Combine:** bash submission.sh combine
- **Upload:** bash submissio.sh upload <path-to-the-file>
- **Total:** bash submission.sh total
- **Update:** bash submission.sh update. Once this is executed, it asks the user to input the student's roll number and name along with the name of exam whose exam we are going to change marks. Atlast it asks for the new mark.
- **Stats:** bash submission.sh stats <operation-to-be-performed>. Once this is entered, it asks for the mark set on which the operation is performed.
- **Show:** bash submission.sh show <roll-number-of-the-student>

- **Display:** bash submission.sh display. Once this is entered in command-line, it asks for what graph to plot and on what range of marks. If you are plotting stats graph, it asks if the mark of any particular student should also be plotted for comparing.
- **List:** bash submission.sh list
- **Git Init:** bash submission.sh git_init <path-to-the-remote-folder>
- **Git Add:** bash submission.sh git_add <file-to-be-added>
- **Git Commit:** bash submission.sh git_commit -m "<Commit message>"
- **Git Checkout:** bash submission.sh git_checkout <Commit-id> (or) bash submission.sh git_checkout -m "<Commit-message>" (or) bash submission.sh git_checkout master
- **Git Log:** bash submission.sh git_log
- **Git Status:** bash submission.sh git_status
- **Git Amend:** bash submission.sh git_amend -m "<Commit-message>"
- **Git Clone:** bash submission.sh git_clone <Path-of-the-git-Repo> <Path-of-output-directory>
- **Git Currrepo:** bash submission.sh git_currrepo
- **Git Switchrepo:** bash submission.sh git_switchrepo <path-to-the-repo>

7 Error Handling

We are Humans. And making mistakes is in our blood. So basic errors like mistyping a command or entering a non valid roll number and so on are being addressed with personalised error messages that can redirect the user to the proper command.

Firstly, if the user doesn't type a valid command, they are redirected to the list command which lists all the available commands.

Next basic mistake is that while entering roll number of a student, it is highly possible that the entered roll number is not a valid roll number. In that case, List of roll numbers are displayed and the user can choose one among the valid ones.

While typing any command, if the format of the command is not how it is specified in Usage of Functions section above, it raises an error conveying the command has invalid arguments and the correct usage is displayed on screen so that the user can use the command correctly next time.

Now, Coming to the git functions, which have a lot of corner cases that needs to be addressed.
In the git_init function, if the directory given as command line argument doesn't exist, then an error is printed saying the directory doesn't exist.

In the git_commit function, an error is raised if you try to commit without initialising any repo.

In the git_checkout function, there is good chance that either the commit id doesn't exist or there could be more than one commit with the starting with same sequence. In either of these cases, the function displays the valid commit ids and asks the user to choose one among the valid ones.

When playing with paths to directories, there could be a possibility that the path doesn't exist or that path doesn't have a git initalised in it already. Such cases are handled by conveying self-written error messages in simple english so that the user can understand what is the issue.

8 References

For developing our bash script and incorporating customizations, we referred to various online resources and documentation. Some of the key references include:

- Bash documentation: `https://www.javatpoint.com/bash`
- Chatgpt for troubleshooting and code examples.
- Python documentation for the Libraries Numpy, Matplotlib, Scipy, Pandas: `https://www.w3schools.com`

9 Conclusion

In conclusion, this bash script offers a comprehensive solution for managing and interpreting CSV files containing student data. Through modularization, extensive commenting, and customization, we have ensured the script's readability, maintainability, and flexibility for future enhancements.

I would like to express my gratitude to Saksham Rathi for providing guidance and support throughout the development of this project. Additionally, I appreciate the valuable insights and feedback received from peers and mentors during the project's implementation.

Thank You..!!