Indian Institute of Technology
Bombay

Department of Computer Science

Bash Grader Report

Mrunal Vairagade
23B1031

Supervised by
Prof. Kameswari Chebrolu & TA Saksham Rathi

April 28, 2024

Contents

1 Objective

This project is intended to create a csv file manager and interpreter. We need to create a bash script that will perform some actions on the CSV files and also have to implement git commands such as git_init, git_commit and git_checkout.

2 Instructions

2.1 Basic Commands

2.1.1 combine

- **Usage:** `bash submission.sh combine`
- This command will run only if main.csv either is empty or is not present.
- This will combine all the non empty csv files(except main.csv) into main.csv which will include marks of students in all exams separated by comma.
- Students who are absent in a particular exam will get 'a' in the corresponding student-exam cell in main.csv.

2.1.2 upload

- **Usage:** `bash submission.sh upload <File_paths>`
- The files will be copied to the working directory and we can run combine again to update our main.csv accordingly.
- Multiple file's path can also be given as command line argument if they are separated by a white blank space.
- If some file can't be found in the given file location this will give an error and won't copy anything.

2.1.3 total

- **Usage:** `bash submission.sh total`
- This will add a new column 'total' in main.csv which will perform total of all marks of a student and append it in the student's row.
- For students who were absent in a exam, their marks in that exam will be taken 0 for totalling.

2.1.4 update

- **Usage:** `bash submission.sh update`
- This will ask for roll number and student's name and if given name doesn't match with the student's name and error will show up.
- Further, if no errors, there will be a continuous prompt for exam name and marks to update and if you want to exit the command type 'exit' in exam prompt.

2.1.5 git_init

- **Usage:** `bash submission.sh git_init <Path of remote directory>`
- It will make a remote directory at the given path specified in second argument or if the remote directory exists it will remove all the content of it.
- Inside remote directory each folder will signify the commit id of commit.
- If this action is action is performed again then the user will be prompted if he wants to reinitialize the git repository to the given path or not. If yes git repository will be reinitialized preserving the commits.

2.1.6 git_commit

- **Usage:** `bash submission.sh git_commit -m ''<commit message>''`
- This part will store all the current version of csv files into folder whose name will be randomly generated 16 digit number (Hash value) as patch files or csv files. This folder will be in the remote repository.
- All these hash values, the corresponding commit messages, commit date and time will be stored in a .git_log file.It will also print modified files, new files and deleted files.
- The commit message shouldn't include '|' in it.

2.1.7 git_checkout

- **Usage:**
  1. `bash submission.sh git_checkout -m ''commit-message''`
  2. `bash submission.sh git_checkout <hash_value>`
- This command reverts our current directory back to the commit having corresponding commit message or hash value.
- Some prefix of the Hash value will also suffice until there are no more than one commit having same prefix. It will give error message.

- If multiple commits are found with same message or starting with same hash value the user will be prompted to select one of those.

2.2 Customization

2.2.1 git_log

- **Usage:** `bash submission.sh git_log`
- This will show a list of commits with their commit messages, commit ids and date time.

2.2.2 show

- **Usage:** `bash submission.sh show <Roll_Numbers>`
- This will show the all the marks for the all the roll numbers given as command line arguments as a pandas Dataframe object.
- We can give multiple roll numbers separated by white blank space as command line arguments.
- If for some given roll number data is not found then it will print "No such roll number" for that roll number and for those whose data is found it will print the marks as a pandas Dataframe object.

2.2.3 statistics

- **Usage:**
  1. `bash submission.sh statistics mean` for mean.
  2. `bash submission.sh statistics median` for median.
  3. `bash submission.sh statistics stddev` for standard deviation.
  4. `bash submission.sh statistics three-fourth` for marks of 75th percentile student.
- This will print the particular statistics of all students in all exams.
- If any other command is run then it will show "Invalid syntax".

2.2.4 grade

- **Usage:**
  1. `bash submission.sh grade -n` for grading wrt number of students in each grade.
  2. `bash submission.sh grade -m` for grading wrt cut off marks of each grade.

- If used -n then user will be asked the number of students in each grade one by one and after entering all main.csv will be updated with a new column named 'grade'.
- If used -m then user will be asked the cut off marks of each grade and main.csv will be updated accordingly.
- Once graded, the user shouldn't try to any operations on main.csv except all git commands and combine as after graded no changes can be done to marks or exams.

2.2.5 graph

- **Usage:**
  1. `bash submission.sh graph -a` for making a bar graph of all statistics of all exams
  2. `bash submission.sh graph -s <Roll Numbers>` for making a graph of all exams of given students.
- Using "-a" will build a graph with subgraphs in it with title exam name. The x axis of graphs will be each statistics function(mean,median etc.) and y axis will be marks.
- Using "-s" will build a graph with subgraphs in it with title exam name. The x axis of graphs will be student's roll numbers and y axis will be marks.

2.2.6 Error Handling

- **upload**
  1. Multiple file's path can also be given as command line argument if they are separated by a white blank space.
  2. If some file can't be found in the given file location this will give an error and won't copy anything.
- If any of the git commands are run with wrong syntax then error message of invalid syntax will show up.
- If git_commit is run without initializing a remote repository an error message of no remote repository will show up.
- If git_init is performed again then user will be prompted if he wants to reinitialize the git repository to the given path or not. If yes git repository will be reinitialized to the specified path preserving the commits.
- Commit in git_commit will may be stored as patch files or csv files depending upon files to commit and files in last commit.

- If we are performing git_checkout and if there are some modifications in the present files wrt last commit an error message will show up and will ask the user to commit the files first to checkout.
- In git_checkout, if no commit is found favourable to given string as commit_id or commit message an error message will pop up.

3 Logic of commands

3.1 combine

I am first copying all the distinct roll numbers along with their names in main.csv by iterating through all non empty csv files(except main.csv) in pwd. Then I am again iterating through all such files to upload marks of students who are present in that exam. In this iteration I am iterating through all students in main.csv and if some student is not present that respective exam.csv, I am appending 'a' instead of marks.

3.2 upload

This will just copy given files to the pwd.

3.3 total

There are two cases in this. First if total is not present in first line of main.csv, total1.awk will run and its output is copied to a random file(here yiha) and then this random file is copied to main.csv and then is deleted. Logic of total1.awk is just iterating for all records with $NR > 1$ (as with NR=1 the column name is listed, here total is appended) and for all records sum is performed from 3rd field in that record(as first two fields contains name and roll number) and this sum is appended at the end of the record.
Secondly if total is already present in first line of main.csv , total2.awk will run and further process of copying will be same. Totalling will also be same except that for each record all but last fields will be printed and after this the sum will be printed.

3.4 update

Only logic to explain is to change the marks in main.csv and that respective exam.csv. I used awk to update the line and that updated line will be replaced by sed. The awk file will have two case one when there is total in first line of main.csv and other in which it isn't. If total is found then we would also need to update the total of the student.

3.5 git_init

The path of remote repo is stored in a .path.txt file which will be hidden in the pwd. Initialization and Re-initialization is typical and don't have much logic but a bunch of cases in case of re-initialization. Also a .git_log file will be made or will be present in the remote repo in all cases.

3.6 git_commit

By running this command the code will first show any modified files, deleted or any new files. For commit a random 16 digit hash number is generated which will act as a commit id and also the commit will be stored in a folder whose name will be that id. Now if the there are no new files and no deletion of files when compared to the HEAD commit patch files[1] will be generated wrt that head commit files and will be stored in that commit folder. If there are new files or some files are deleted then all csv files are copied to the commit folder. In this case this commit will act as a reference commit for further commits until a new reference commit is made. The information whether the commit is reference commit or not will be stored as a string in .git_log in the line where the commit is written. There is a HEAD commit too which will show which commit you are presently in. This information is also stored in .git_log file as a string "head" in the commit line.
The syntax of each line of .git_log is

$$\text{“}commit\_id\text{”}|\text{“}commit\_message\text{”}|date\_time|(;;) \quad head$$

Here the strings "(;;)" and "head" may or may not be present in each line. Everytime there will only be one head in .git_log.
Also to note, the commit message shouldn't include '|' in it.

3.7 git_checkout

Checkout will also be based on two cases, one where the given commit is the reference commit and other where the given commit isn't a ref commit. In first case, the command will remove all csv files in pwd and copy all csv files to pwd. In .git_log the position of head will set to this commit line and other head will be erased of. In other one, the files are patched wrt the latest reference commit made before the checkout commit. The details of this are explained more in checkout.sh.

3.8 Other Customizations

git_log will print commit info in such a order that first commit will be HEAD commit and it will appear first. Also some colour of fonts is also set[2]. These are more typical and most of the logic is covered in lectures of course.

4 Utilities

The programming languages used in this project by me are:

- Bash[5]
- Sed & Awk
- Basic Python
- Python Libraries
  - Numpy[5]
  - Matplotlib[3]
  - math
  - statistics
  - pandas[4]

References

[1] https://www.scaler.com/topics/linux-patch/#.

[2] https://stackoverflow.com/questions/5947742/how-to-change-the-output-color-of-echo-in-linux.

[3] https://www.w3schools.com/python/matplotlib_intro.asp.

[4] https://pandas.pydata.org/docs/.

[5] Prof. Kameshwari Chebrolu. Lecture notes, 2024.