Physics 129L: Problem Set 2

Zihang Wang

October 2023

Problem Set Submission Guideline

A) Github Submissions Starting with problem set 2, we will use GitHub for problem set submissions. To access the problem set, please **fork** and **clone** the **forked** repository to your local virtual machine. **Please complete the problem set in this forked directory.** Submit **a pull request** for merging into the main branch before the problem set due date.

B) Export Command History: Please submit all of your commands (including errors) in a text file named "history.txt" in the problem set directory. For instance, to select and append the last 10 commands to the file, use the following history command: "history | tail -n 10 >> history.txt". Note that the last command in the text file will be the history command mentioned above.

C) .tar.gz File compression and submission on Github for each problem set, you are asked to submit the compress version of the problem set to github via git operation. Here is a step by step guideline,

1. Use the **tar** command to compress the problem set directory into a **single** ".tar.gz" file.
2. Obtain the sha256sum by running "sha256sum P2.tar.gz".
3. Echo the **full sha256sum** to a text file named "sha25sum_problem_set.txt".
4. Initialize a git repository named"Archive_P# (#: problem set number) on your local machine, and move both the "tar.gz" file and the "sha25sum_problem_set.txt" file to the repository.
5. Create an empty **public** directory udner the **same name** in **your own github account**.
6. **Push** this local repository to the remote repository.

In summary, by the end of the due date, you should have two directories: one is the **forked directory, where you submit your pull request**, and the other is **your own archive directory**. In this problem set, you are also asked to submit your problem set 1 archive to the github (see **problem 3**).

Problem 1: File Management and Bash Scripts

Bash scripts hold significant value in modern Linux system administration; they are used for automating tasks and performing various operations. In this problem, we will explore the fundamental uses of bash scripts.

In modern computing, file searching is critical for data extraction. The output data files from an instrument are typically labeled with information such as subject, year, month, date, and index (if there are multiple files). For example, in an electron scattering experiment, the instrument produces data files as binary files with names like:

$$\text{example\_file\_name}: \quad \text{electron\_scattering\_}2023-10-04\text{\_sample\_index\_}0.\text{bin}$$

In this question, you are will manage the data files created by your collaborator who measure the energy lost in an inelastic electron scattering experiment.

1. Fork the problem set /P_2 on GitHub, and then clone the forked repository. The directory tree has the following structure:

```
/
└── P2
    └── /Problem_1
        └── /electron scattering data
```

2. Imagine your collaborator accidentally output data files to the wrong directory, i.e. /Problem_1. There are 500 files of them, and you need to delete them. **write a **single** line on the terminal that does the job.**

3. Your future collaborators may need to address this issue again, but they may have no knowledge of Linux commands. Write a bash script that defines an alias named 'file_remove.' It should accept a single argument: the name of the target directory. Hint: Use your previous work.

4. Navigate to the directory /electron_scattering_data. Your collaborator has requested that you separate the files with odd indexes from those with even indexes and move them to two new directories within the current directory. The two directories should be named /electron_scattering_data/odd and /electron_scattering_data/even, respectively. Write a bash script to accomplish this task. Hint: Use a for loop.

In three weeks, you will be asked to decode the binary files above with **python** and infer the corresponding statistical distributions.

Problem 2: Number Conversions and Bash Scripts

Although less common, people use bash scripts to perform basic calculations or conversions. There are three types of numbers: decimal (base-10), binary (base-2), and hexadecimal (base-16). An example conversion is provided in

Hexadecimal	Decimal	Binary
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111
10	16	00010000
1A	26	00011010
1F	31	00011111
20	32	00100000

Table 1: Examples on Hexadecimal, Decimal, and Binary Number Conversions

Table 1. In this problem, you are asked to **write a bash script that converts a given decimal number to both hexadecimal and binary numbers. In addition, the program should only accept decimal values smaller than 100000. You are required to hard-code the conversions using information from both Table. 1 and Table. 2!** Please place your bash script into a directory named /Problem_2 within the current directory.

Problem 3: Basics on Version Control with Git.

In this problem, you are asked to create your own repository with git and push your first problem set, the ".tar.gz" file, to the remote as an archive.

1. Create a separate directory, /Archive_P1, and initialize git. (please do not create this directory under /P_2).
2. Copy your first problem set, tar.gz file, to this repository.
3. Initialize the main branch, stage the change, and commit the change.
4. Following the discussion in the lecture, create your personal access token on github.
5. Commit to the remote repository. Hint: you need to first create the remote repository (with the same name) on github!

Division	Quotient	Remainder
125064 / 16	7816	8
7816 / 16	488	8
488 / 16	30	8
30 / 16	1	14
1 / 16	0	1

Table 2: Example: converting Decimal to hexadecimal via division remainders. In hexadecimal, it is 1E888.Or in binary, it is 11110100010001000.

Problem 4: Version Control with Git.

1. Checkout a different branch, called feature_branch. Create a readme file with "hello world!". Then, stage, commit and push to the **feature_branch** and push to the remote.

2. After you push the feature_branch, merge the feature_branch with the main branch locally, and push to the remote. What do you see locally and remotely on github?

3. Modify the text in readme file from "hello world!" to "hello world from remote!" **on github manually**, and pull from the remote origin. Do you see the update?

Problem 5: Archive your problem set 2.

Please follow Problem 3, but now you need to do it for problem set 2.

Problem 6: Pull request.

Submit a pull request on github. I will later review your code and give you comments on your submission.