2. You can come up with your own user interface. For user Interface we are not expecting anything beyond command line, but you are welcome to create other exciting interfaces.

IX. ASSIGNMENT SHELL

Making a command line app which helps you manage and maintain the file structure on your assignment files, as well as readily accept updates to the problem structure.

Assigned to: ***Tushar Joshi***

A. Legend

After the assignment is released, TAs make changes to the problems, update the PDFs and the distribution packages, their file system checker code does not work and we have to send them screenshots for debugging, it's all so annoying. Let's go fix this process.

B. Commands Spec

Your shell should work like normal bash, show a prompt like: `animesh/dsa>` , where you can type your commands. The following commands need to be supported. The abbreviated name of the subject you are currently working on should be shown in the shell. This should persist across runs.

1. `switch <subject>`: Change the name of the subject in the prompt and change folders appropriately.

2. `create <assignment>`: Creates a new folder for the assignment, downloads (or copies locally) the contents of the dist folder and the problem statement into the current directory.

3. `update <assignment>`: Downloads (or copies locally) the new assignment files, deletes the old ones, and replaces them.

4. `setup <assignment>` (optional): Reads an indented text file which has the folder structure to be followed. Creates that folder structure.

5. `test <assignment>`: Runs the submitter.py file in the dist folder, if it exists. Store the logs, be it compilation error or others in a file that can be sent to the TA for debugging.

6. `submit <assignment>` Makes a zip file of the current directory. Uploads the file to our servers (or copies to local directory)

7. `compare <assignment> <zipfile>` Compares the file tree to check if any of the files in the submitted zip are different from any of the files in the current directory. Prints list of those files. Uses MD5 hash (command MD5 sum) to compare.

8. `use <assignment>` Changes the prompt to `animesh/dsa/<assignment>` and for all commands now, if the `<assignment>` argument is not passed, it will default to this assignment.

C. Tasks List

1. Make a basic interactive shell (Shameless plug of pretty code for interactive shell: https://github.com/AnimeshSinha1309/os-assignments/tree/master/AniShell - only see small parts of processor/prompt.c, processor/parser.c, `processor/tokenizer.c`). You don't need to implement any of the features here, just taking input, splitting by spaces, and making cases for each command.

2. Make the commands as listed in the spec above. Try to write basic tests to see if all the commands are working (optional). Each command will carry some marks, and this is the bulk of the project, so build incrementally.

3. Propose what commands need to be added to make the download and running of these in evals (the way we did for assignment 1) as well as running MOSS becomes very easy and fast. Suggest how you will implement these features. Can you ensure that there are almost no cases where the students say "It was working, but now it's not?", if we give many of the test cases. You don't need to implement anything here.

D. Comments on Implementation

You might need to implement a stack to store and parse your inputs. You may also need to maintain a tree to represent the current directory structure, find changes, updates files, etc. Try to take your best call on these. This project is implementation heavy. We would appreciate well handled error cases, and nice pretty code.

X. DIRECTORY MANAGER

You need to develop a directory manager that allows to maintain and search directories efficiently.

Assigned to: ***Divanshi Gupta***