CLI-1 Task

- Create the following directory structure. (Create empty files where necessary)

```
hello
├── five
│   └── six
│       ├── c.txt
│       └── seven
│           └── error.log
└── one
    ├── a.txt
    ├── b.txt
    └── two
        ├── d.txt
        └── three
            ├── e.txt
            └── four
                └── access.log
```

So, here I need to create the above tree structure of the directory. Below are the steps and explanations for the above task.

step1) First, looking at the diagram I need to identify whether it is a directory or a file. Here there is a parent directory named “hello” so I’m going to create a directory “hello” with the command:

> **mkdir hello** (mkdir directory-name)

Here, the “mkdir” command is used to create the directories.

step2) now we need to create two directories “one” and “five”, before that we need to be in the hello directory. So we use the command “cd” to change the directory

> **cd hello** (cd directory-name)

Here, I change the directory and move it to the hello folder.

> **mkdir {one,five)**   (mkdir {dir1,dir2})

Here, I created 2 directories one and five as shown in the tree structure with the help of creating multiple directories using the mkdir command, mkdir and in curly braces write the directory name separated with “,” to create multiple directories with one command.

step3) change the directory to either one and create the rest of the tree structure.

> **cd one**
> **touch a.txt**   (touch file-name)
> **touch b.txt**
> **mkdir two**

Here “touch” command is used to create an empty text file, we can also use “cat > file_name” to create the file.

step4) change the directory to two and create the rest structure.

> **cd two**
> **touch d.txt**
> **mkdir three**
> **cd three**
> **touch e.txt**
> **mkdir four**
> **cd four**
> **touch access.log**

Above are the similar commands which are used to create the rest of the structure in folder two.

step5) now the structure for folder one is created now we need to do the same for folder five. For that we need to revert back to the original folder position which is in (./hello/) and are in dir four which is in (./hello/one/two/three/four) we need to go all the way back to the (./hello) dir, for that, we will use command (cd ..)

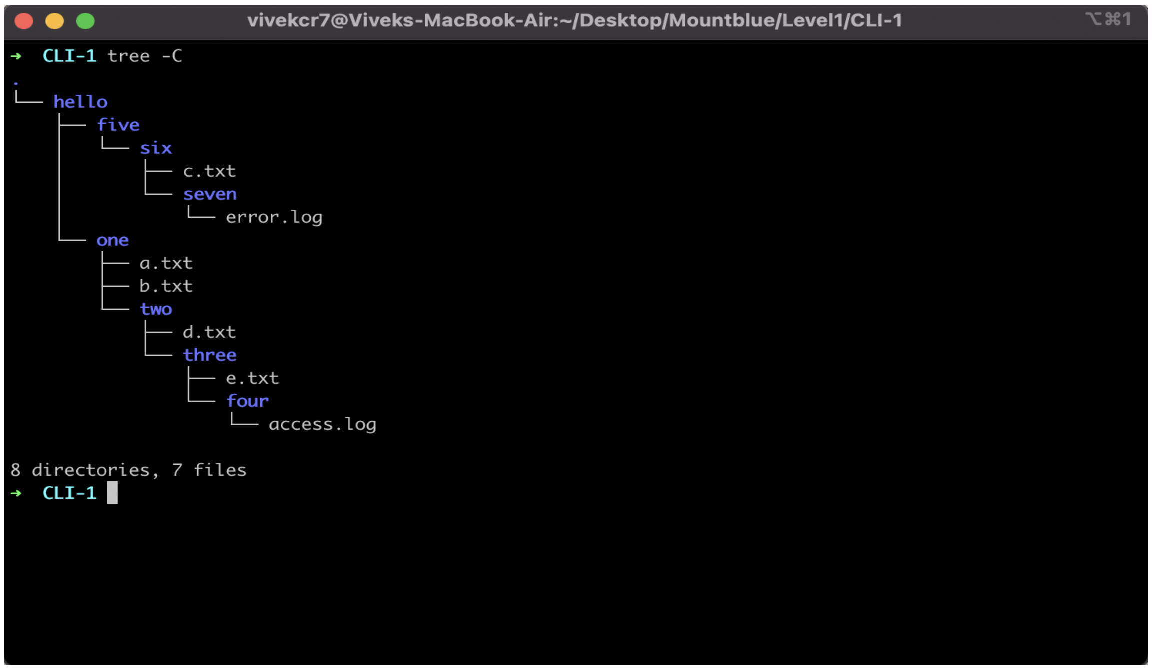
> **cd . .**

Repeat it till we get back to the hello folder.

step6) now change the directory to “five” and create the structure as shown above.

> cd five
> mkdir six
> cd six
> touch c.txt
> mkdir seven
> cd seven
> touch error.log

I think we have created the structure. The below fig shows my local systems directory.



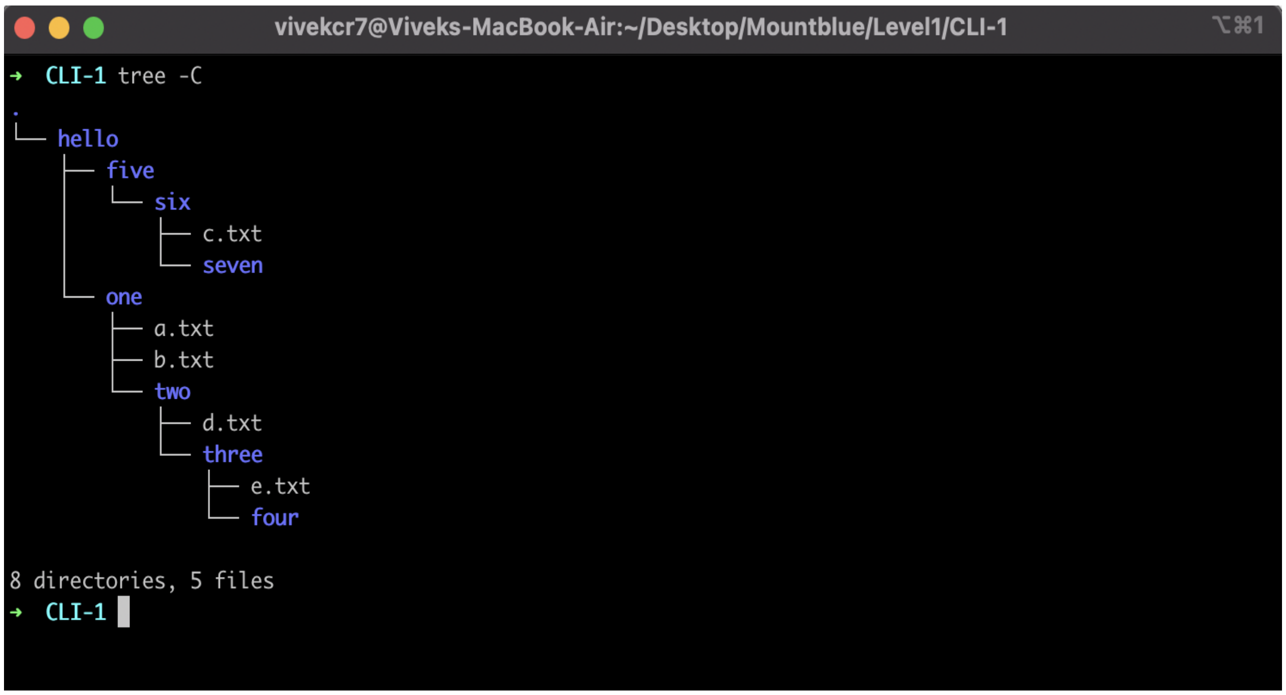
```
vivekcr7@Viveks-MacBook-Air:~/Desktop/Mountblue/Level1/CLI-1
➜  CLI-1 tree -C
.
└── hello
    ├── five
    │   └── six
    │       ├── c.txt
    │       └── seven
    │           └── error.log
    └── one
        ├── a.txt
        ├── b.txt
        └── two
            ├── d.txt
            └── three
                ├── e.txt
                └── four
                    └── access.log

8 directories, 7 files
➜  CLI-1
```

- Delete all the files having the .log extension with one command.

> rm **/*.log

Here the (**/) helps to match the files in the current directory and its subdirectories recursively.
(/*.log) will normally delete all the files with the extension (.log) run the tree command and see according to the above tree structure (.log) file got deleted



```
vivekcr7@Viveks-MacBook-Air:~/Desktop/Mountblue/Level1/CLI-1    ⌥⌘1
➜  CLI-1 tree -C
.
└── hello
    ├── five
    │   └── six
    │       ├── c.txt
    │       └── seven
    └── one
        ├── a.txt
        ├── b.txt
        └── two
            ├── d.txt
            └── three
                ├── e.txt
                └── four

8 directories, 5 files
➜  CLI-1 ▌
```

- Add the following content to a.txt.

Unix is a family of multitasking, multiuser computer operating systems that derive from the original AT&T Unix, developed starting in the 1970s at the Bell Labs research centre by Ken Thompson, Dennis Ritchie, and others

Here we need to add the above content in an a.txt file.

> **echo "above-content" >> a.txt**

> **cat a.txt**

Here I used (echo "content" >> file-name) echo is used to print for example and the strictly greater than (>>) is used to append the echo text at the end of the file it is best practice to use (>>), it can also be done by (>) single greater than operator but it will overwrite the current data so choose wisely. And (cat) command is used to display the content of the file in the terminal itself.

- Delete the directory named five.

> **rmdir five (if the directory is empty)**

> **rm -rf five (if the directory is not empty)**

rmdir command is used to remove directory but only if the directory is empty, in our case the directory has subdirectories so instead of "rmdir" we going to use "rm -rf dir-name" to delete the directory with its contents.
(**rm** means to remove and **r** means recursive since we have to use entire folder and its subfolder **f** means force removal (**-rf**) means the recursive removal)

- Rename the `one` directory to `uno`.

> **mv ./one ./uno**

Here we just need to rename the directory "one" to "uno" so I used the "mv" command to rename it is also used to move the file. (mv <source-directory> <target-directory>)

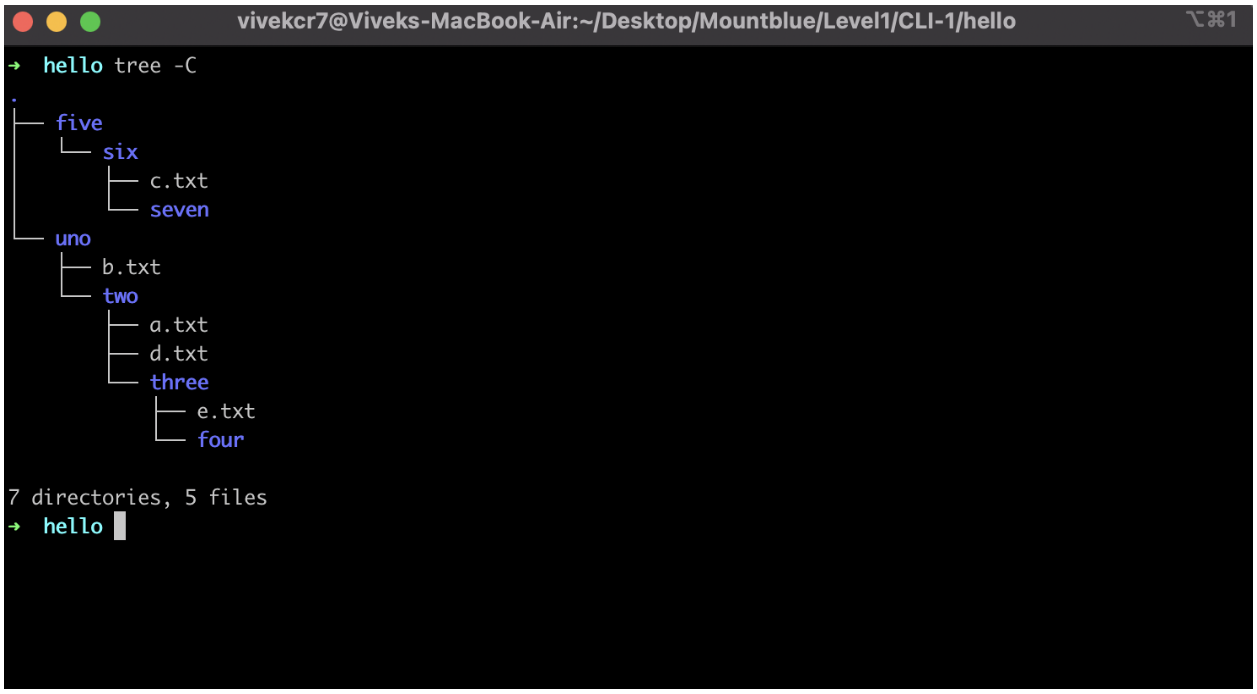
> ls

To check, use the “ls” command, it will show all the dir and files of the current directory.

- Move `a.txt` to the `two` directory.

> mv ./a.txt ./two

Here I used the “mv” command to move the “a.txt” file to folder “two” to view result type command “tree”



```
vivekcr7@Viveks-MacBook-Air:~/Desktop/Mountblue/Level1/CLI-1/hello    ⌥⌘1
➜  hello tree -C
.
├── five
│   └── six
│       ├── c.txt
│       └── seven
└── uno
    ├── b.txt
    └── two
        ├── a.txt
        ├── d.txt
        └── three
            ├── e.txt
            └── four

7 directories, 5 files
➜  hello 
```