Jay Kmetz
800249366
CS465 Programming Assignment 2
Due 04/28/2020

Status

The program works entirely and has an extra functionality of manual_control mode. The manual_control mode was initially there for testing but I liked it so I kept it in.

Programming Environment

- Python 3.5.2
    - External Imports
        - Import os
        - Import sys
        - From os import path
    - Internal Imports
        - Import code.constants as CONST
        - From code.classes import *
        - From code.commands import *
        - From code.helpers import *

Detailed Description of Algorithms and Such

Main Structure:

The entry point for this program is in access.py. The main command runs file_genocide() which destroys any trace of previous executions, and then enters either manual_control mode (if the user types in manual_control) or file mode where all lines from a file are executed. Each of these modes utilize the run_command method which keeps the session info in global scope, splits the received text into the command and a param string, searches the cmdList dict made in commands.py based on the command, and then executes the correct function from that list with either the split parameters or "" if there were no parameters. It also passes in the session so the commands can modify that.

There are three main classes which allow this program to run the way it does: File, Group, and Session. The File class will be initialized each time a file is created and stores its permission, name, owner, and group. It also has its own toString method which allows the user to print out all information necessary about the file. This is used by ls and at the end it is used to list out files in files.txt. The Group class has a name attribute which keeps track of its name and also a users attribute which keeps track of the users in it. It also has one method which takes in a user and returns true if that user exists in that group. The final class is the one that ties these two together: Session. Session keeps track of its current user with the cu attribute, the files with a files dict which maps file names to File classes, the groups with a dict that maps group names to Group classes, and users which keeps tracks of all users created in the system.

Code Functionality:

Going into commands.py, there is a list of functions whose name corresponds to the typed in command. At the bottom, you will see a dict called cmdList which maps all of the commands to their respective function. This is what gets called in access.py. In this way, everything is modularized. Each of the commands work off of the same principle. Error check first, and if there are no errors, perform the action. Each command is built piece by piece with each piece adaptable. This is the way code should be made.

Going into helpers.py we see a few functions that help us along the way such as login account which searches the ACCOUNTSFILE for the password of the user trying to log in. This would be more in depth if it had to do password hashing but I will take the way that the project description specifies and only compare ascii values. There is a dualLog function which is very helpful as it logs the same message into the console and into the LOGFILE. There is file_genocide which was explained earlier. There is a strToPerm function which converts a string such as "rw- rw- r--" to the correct hexidecimal number that permission represents for easy conversions, and there is an executionEnd function which goes through and writes files.txt and groups.txt. All file streams should be closed at this point because every file access was made with the "with" keyword.

In constants.py resides various constants and filepaths/names which the program needs to use constantly.

There exist a few __init__.py files which make importing possible.

Documentation and Outputs:

All documentation (including this pdf) is stored in the documents folder. Files are stored in the files directory for organized storage and all output files are stored in the outputs folder. This keeps things clean and it doesn't muck up the root directory. Any code that is working behind the scenes is stored in the code folder. The classes are strored in the code/classes folder. There exist two test case folders jtc1 and jtc2 which hold J's test cases 1 and 2. The input files are jtc1INPUT.txt and jtc2INPUT.txt. The outputs and files folders that were generated from running "python3 access.py jtc1INPUT.txt" or "python3 access.py jtc2INPUT.txt" are stored in their respective files and outputs folders. (To be specific, jtc1/files, jtc1/outputs, jtc2/files, jtc2/outputs). If these were to be tested, type in one of the test cases listed in the quotations above and the outputs and files folders should have the same contents as the ones stored in their respective folders.