Ashwin Subramaniam
AXS190172

Project 3 Design Document

Phase 1

UM.exe

- The UM has an array of User struct which stores user id, pid, socket, login state etc.
- Once the UM is run it listens to the given socket and accepts new clients.
- Once a new client is accepted a separate thread and shsh process is created per user and the User struct is updated with the new user's ID, pid, socket and login state is set to true.
- A new directory is created for each user where she can work with commands.
- Incoming ucmd input is parsed and executed by shsh.
- A temporary file called temp.out is used where the shsh will write the output of the given command temporarily which will be read by the socketThread method called by the pthread_create and this output is written back to the client
- If a logout is given, the specific user's process and the corresponding thread is also ended

Client.exe

- Once the client connects to the server it indefinitely prompts user for ucmd>
- It first writes the command received from screen to the server
- It then reads the response from the server and displays on screen

Phase 2

Admin.exe

- The UM will print its PID as soon as it starts. The admin should be started with this PID as command line argument.
- The admin sends one of the 4 Signals to the UM server using the kill command using UM pid
- The admin runs until terminate is called

UM.exe

- The UM uses signal() to catch the kill command from the admin
- Each signal has corresponding methods to handle sleep, infor, listuser, terminate
- The global User structure is accessible within the signal handler methods
- *Listuser* iterates through the list of users in the user struct and checks for the login state and displays the users who are active
- *Infor* method makes use of globally declared port no, etc., and iterates through the list of users to find the active ones.
- *Sleep* method puts the UM to sleep but lets the client processes work fine. Once sleep duration is completed UM comes back with message 'Sleep completed..'
- *Terminate* method loops through all active users using the user struct and exits active threads, and sends a kill SIGTERM to all the userstruct[].pid and exits gracefully from the UM