Operating Systems, Assignment 5

This assignment includes a programming problem in C. As always, be sure your code compiles and runs on the EOS Linux machines before you submit. Also, remember that your work needs to be done individually and that your programs need to be commented and consistently indented. You **must** document your sources and, if you use sources, you must still write at least half your code yourself from scratch.

1. Teachers of North Carolina State University are having a busy schedule in this Thanksgiving month and their schedules are frequently overlapping. That's why In this assignment we are going to implement a multi threaded server/client Class Room booking system in C that uses TCP/IP socket mechanism for communicating. To handle synchronization you have to integrate appropriate code (i.e POSIX monitor).

   For this program it is sufficient to write a server program that will be handling requests **classRoom-Scheduler.c**. No need for implementing client side program (Teachers), we are going to use a general purpose program for network communication , `nc`. Note that in this assignment we are referring Teacher as the client.

   Once you get your server program running it will keep on handling incoming requests from the Teachers (Clients). For connecting to the server the Teacher will need the hostname (which is localhost in our case) and a port number (54321 an arbitrary value defined in the static section of the code).

   Our solution is pretty straight forward. There will be a number of classrooms initialized in the beginning (say 5) with some specific names (i.e ROOM1201). A Teacher can book a class room if it is available. When he is done he can leave the room making it available for another Teacher. If a class room is taken by a Teacher and another Teacher wants it, he has to wait unfortunately until the room is vacant again.

   While a Teacher is connected to the system, the server will repeatedly prompt for a command using the prompt "`> `". We are simply echoing whatever the Teacher types in the skeleton code. However in the final version of it you need to implement some specific commands that corresponds to the following actions:

   - `book` *Class Room*
     Wait until it's possible to book a room with the given name, if it is booked, remove it from the listed available rooms. After booking a room a proper message feedback should be given to the Teacher (i.e Class Room Booked!). If an invalid room name is given proper feedback should be also given(i.e Class Room does not exist).
   - `vacant` *Class Room*
     Takes a Class Room Name as argument and vacates the class room with the given name (i.e., add it to the set of available room). If any other Teacher was waiting to book this class room, this should wake up one of them up so they can book it. If invalid room name is given proper feedback should be given to the Teacher.
   - `available`
     Lists out the available rooms.
   - `quit`
     This is a request for the server to terminate the connection with this client. This behavior has already been implemented for you.