Network Management Application using the Sockets API

The focus of the projects in this course is to be able to compare, in quantitative and qualitative terms, the two approaches commonly used in the development of distributed applications. Project 1 deals with the first approach, i.e., the socket based approach. Project 2 deals with the RPC/RMI based approach. The paper that you will write at the end of these projects will compare these two approaches in quantitative and qualitative terms. You will be required to measure the mean response time using the socket and the RPC/RMI based approaches.

This client-server programming project is a concurrent TCP based network administration tool. The server provides current system status as requested from the client. The client can request from the server such information as the system time, the uptime, the memory use, netstat, current users, and the running processes. The client program displays a text menu for the user. The user makes requests by selecting a menu option.

The user will enter the server hostname as a command line argument when the client program is invoked. If there is no command line argument then the program will print an error message and exit. The client program then enters a loop until told to quit where it will:

- Display a menu
- Prompt the user for a command
- Test user input for command validity. If user command is invalid, inform the user and redisplay the menu.
- Send that command request to the server on the host
- Get response back from server
- Display response for user

The menu will provide the following choices to the user:

1. Host current Date and Time
2. Host uptime
3. Host memory use
4. Host Netstat
5. Host current users
6. Host running processes
7. Quit

Make your server concurrent by having it fork a process to handle each client request. Have your server print out diagnostic messages about what it is doing (e.g. forking a child process, accepting a new connection, etc). Also measure the mean response time.

You may use the popen() library function to get the output of a command into your program. Here?s a code fragment to experiment with and see how popen() works:

```
char buf[100];
FILE *fp = popen("who", "r");

while (fgets(buf, 100, fp) != NULL)
        printf("%s\n", buf);
```

You will be required to provide a demo to the instructor. The client-server application will be tested for compliance with all requirements listed above. Your code will be expected to deal with invalid user commands. You will turn in binaries and source code of your client and server programs at the time of the demo.