CSCE 3600: Systems Programming

Major Assignment 2 – Sockets & Synchronization

Due: 11:59 PM on Wednesday, December 7, 2016

COLLABORATION:

You should complete this assignment as a group assignment with the other members of your group. Each group should have 3 or 4 members, no more, no less. Submit only ONE program per group. Also, make sure that you list the names of all group members who participated in this assignment in order for each to get credit.

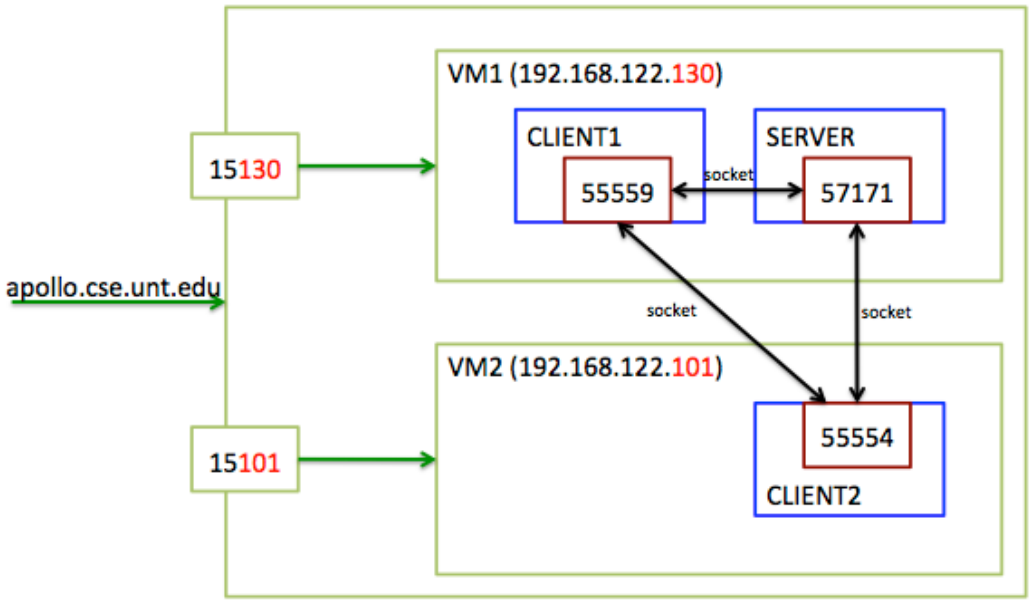
PROGRAM DESCRIPTION:

In this assignment, you will write a complete C program to support a client/server and client/client model using Linux sockets for a "census" system using the `apollo` server instead of the standard Linux CSE machines (i.e., `cse01` – `cse06`). The program will consist of a "main" recipient of the census data (i.e., the server) that will keep a running sum of all the census data being sent by two regional agencies (i.e., the clients) as follows:

- **Server**
  - The server will receive integer values from either of the two clients (in any order) and add these values to its current total, which the server then returns back to the originating client so that the client is aware of the most up-to-date "census" total. Unlike *Minor Assignment 7*, these values sent in from the clients will be input manually from the keyboard at each client.
  - The server will continue to "listen" for clients and run indefinitely until manually terminated (using, for example <CTRL-C>).
- **Clients** (note that there are 2 clients)
  - A user will manually enter integer values at each client that will be sent to the server. The server, in response to this integer value, will send the client a current total so that the client is aware of the most up-to-date "census" total.
  - Each client will continuously monitor the CPU usage of its machine and if the CPU usage reaches or surpasses a threshold value specified by the user, the client will (1) send the sum of the integer values sent by the other client to the other client, and then (2) disconnect from the server. The remaining "active" client may continue sending its own integer values to the server.
  - A client may voluntarily "quit" sending integer values to the server when the user manually enters a 0 for its integer value (whereupon the client will disconnect from the server).

APOLLO SERVER:

Each group will be assigned two VMs on the **apollo** server according to your group number on Blackboard. The server and client 1 will be run on one of the VMs, while client 2 will be run on the other (but client 1 and client 2 will consist of the same code). The configuration for each team will be similar to the following diagram, although IP addresses and ports will vary for each team:



After connecting with VPN, each group will connect to the **apollo** server address, `apollo.cse.unt.edu`, using their assigned ports, which will be port forwarded to their specific VM. Then, once on the respective machine, you will invoke your programs for the server and clients in the following manner (using the above diagram as reference):

SERVER:

server <svr_port>

For example, `server 57171`.

CLIENT 1:

client1 <rem_ipaddr> <svr_port> <cli1_port> <cli2_port> <cpu_%>

For example, `client1 192.168.122.101 57171 55559 55554 20`.

CLIENT 2:

client2 <rem_ipaddr> <svr_port> <cli1_port> <cli2_port> <cpu_%>

For example, `client2 192.168.122.130 57171 55559 55554 15`.

REQUIREMENTS:

- Your code should be well documented in terms of comments. For example, good comments in general consist of a header (with your name, course section, date, and brief description), comments for each variable, and commented blocks of code.
- Your two programs should be named "**`svrMajor2.c`**" and "**`cliMajor2.c`**", without the quotes, for the server and client code, respectively.
- To ensure that your C code is compiled correctly, you may either include a `README` file with instructions on how to compile your code or a `Makefile` so that our scripts can just run `make` to compile your code with the correct libraries and flags.
- Your program will be graded based largely on whether it works correctly on the **`apollo`** machine (actually, your assigned VMs), so you should make sure that your program compiles and runs on this machine.
- Please pay attention to the **SAMPLE OUTPUT** for how this program is expected to work. If you have any questions about this, please contact your instructor, TAs, or IA assigned to this course to ensure you understand these directions.

SAMPLE OUTPUT (user input shown in **bold green**):

==> SERVER

```
group30@csce3600:~$ ./server 57171
Waiting for Incoming Connections...
Client Connection Accepted
Client Handler Assigned
Client Connection Accepted
Client Handler Assigned
3
13
33
38
68
Client Disconnected
75
78
86
Client Disconnected
```

==> CLIENT 1

```
group30@csce3600:~$ ./client1 192.168.122.101 57171 55559 55554 10
Connected
Enter CLIENT 1 Data: 3
SERVER Total: 3
Enter CLIENT 1 Data: 5
SERVER Total: 38
```

```
Enter CLIENT 1 Data: Connection Accepted
Handler Assigned
Received 3 from Other Client
Client Disconnected
7
SERVER Total: 75
Enter CLIENT 1 Data: 3
SERVER Total: 78
Enter CLIENT 1 Data: 8
SERVER Total: 86
Enter CLIENT 1 Data: 0
CLIENT 1 Disconnecting...
```

==> CLIENT 2

```
group30@csce3600:~$ ./client2 192.168.122.130 57171 55559 55554 8
Connected
Enter CLIENT 2 Data: 10
SERVER Total: 13
Enter CLIENT 2 Data: 20
SERVER Total: 33
Enter CLIENT 2 Data: 30
CPU Utilization: 10.00%. Threshold 8% Exceeded
Sending 3 to CLIENT 1
CLIENT 2 Disconnecting...
```

NOTE: In this sample output, Client 2 disconnected before Client 1.

SUBMISSION:

- You will electronically submit your two C source code files and the `README` file or `Makefile` to the **Major Assignment 2** dropbox in Blackboard by the due date.