BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
IS F462 : Network Programming
I Semester 2014-15
Assignment-2
Weightage: 10% (15M) Due Date of Submission: 23-Nov-2014

==========

Important to Note:

1. Groups of 3 students utmost. Assignment-1 groups will continue by default. If there is a change, inform group info latest by 15th Nov by sending mail to isf462@gmail.com with subject Assignment2-group-info.
2. **Don't use temporary files and system() function.**
3. **Only working programs will be evaluated. If there are compilation errors, it will not be evaluated.**
4. *Provide makefile for each problem.*
5. For any clarifications please contact me (khari@pilani.bits-pilani.ac.in).

Plagiarism will be thoroughly penalized.

==========

P1. Consider a chat server. It is a concurrent server providing concurrency through prethreading. Server takes thread pool size N on the command line. Threads wait on mutex and then call accept(). Client is a telnet application. Client can send the following messages. All messages start with a 4 letter command, followed by text and ended by \r\n. Text is interpreted as per the command.

- *JOIN <name>\r\n*: This is the first message sent by the client providing its own name.
- *LIST*: This message will fetch all the connected online user names.
- *UMSG <tname>\r\n <msg>\r\n*: This is the message used for sending message to particular person named *tname*. If such a person is online then message will be delivered, otherwise server sends ERROR <not online>.
- *BMSG <msg>\r\n*: This message will deliver *msg* to all the online users.
- *LEAV\r\n*: This message will make the server remove the client entry in its data structures and close the connection.

Threads should not do busy-wait. They should use condition variables wherever waiting/notification is required. All shared data must be protected.
Implement prethreads_chatserver.c as per above requirements.

[6M]

P2. Consider a chat server. It is a event-driven concurrent server that receives IO notifications through epoll API. All sockets are set to be non-blocking. Message queues are used as FIFO queue to queue the events. Every request goes through at most 3 events: reading, processing, writing.

Main thread: waits on epoll_wait(). When epoll_wait() returns, it reads each event in the ready list and adds messages consisting of fd, and the corresponding event to the message

queue. At the start up, when main thread starts it adds listening socket to interest list of epoll().
Another thread: Use a separate thread for processing event messages in the message queue. It waits on recvmsg() of message queue. It reads a message. If it is a read event message or write event message, it will read/write to the socket and add further events to the message queue or update interest list of epoll. If the message is a processing event message, it will do the necessary processing and updating of data structures and add more events to the message queue and update interest list of epoll.

Client is a telnet application. Client can send the following messages. All messages start with a 4 letter command, followed by text and ended by \r\n. Text is interpreted as per the command.

- *JOIN <name>\r\n*: This is the first message sent by the client providing its own name.
- *LIST*: This message will fetch all the connected online user names.
- *UMSG <tname>\r\n <msg>\r\n*: This is the message used for sending message to particular person named *tname*. If such a person is online then message will be delivered, otherwise server sends ERROR <not online>.
- *BMSG <msg>\r\n*: This message will deliver *msg* to all the online users.
- *LEAV\r\n*: This message will make the server remove the client entry in its data structures and close the connection.

Server should not get blocked on any IO request. Implement eventdriven_chatserver.c as per above requirements.

[9M]

How to upload?

- Make a directory for each problem like P1, P2 etc and copy your source files into these directories.
- Tar all of them into idno1_idno2_idno3_ass2.tar
- Upload it on http://nalanda.

===End of assignment===