Project 3 – A simple e-mail server

In this project you will design and implement a very simple e-mail server, along with its clients. You need to use TCP socket programming in C. You should design your programs using the knowledge you have gained from projects 1 and 2, and are free to "borrow" code from your first two projects. Just document which code you borrowed. You must borrow only you own code. If you borrow someone else's code that is not called borrowing, it is called cheating.

Your code should run on both zeus and thor.

It works as follows:

1. The server is called "mailserver". The server listens on port number 5000 + X, where X is the last three digits of your GMU G number. When the server starts it prints out the message "Welcome to Bob's E-mail Server, running on port 5xyz." Replace "Bob" with your name and 5xyz with whatever port number you are using. You may hardcode 5xyz into your code.
2. Anytime a client connects to the server the client receives the above welcome message. The client prints out this message. The client then sends its own user name. The client program, called "mailclient," is called using the following syntax:

   mailclient username server_ip_address 5xyz,

   where the username is the name of the user, the server_ip_address is where the server is running, and 5xyz is the port number you are using.
3. Clients send messages to the server using the following format:

   "simon@ipaddress This is the message"

   This means that the email message is meant for user simon on whatever IP address simon gets his email on. *Note that this ipaddress does not have to be the same ip address that either the client or the server runs on.* The actual message does not have to be longer than 80 bytes. This message is given to the mailclient by simply typing it in.
4. The server keeps track of all clients connected to it, and stores all the messages that it receives from the client.
5. Every 30 seconds the server looks at the messages it has stored, along with name of the receiptant and the ipaddress. If the user and IP address is currently connected, the server sends the message to that client and deletes the message. This means that you have to setup a timer, handle the event of the timer being fired, and reset it.
6. The server must also accept a *list* command, which prints out the list of all currently stored messages and their intended recipients, along with a list of currently connected clients.
7. The client must also accept a *close* command, which disconnects from the server and ends their process.

We will test your code using the server and at least 5 different clients running on different machines. Make sure you program works on both zeus and thor.

Hand in your mailserver.c and mailclient.c code, along with any .h files or other .c files you use, along with instructions as to how to compile your code

DUE: NOON, Friday December 6th.