Homework Assignment 01
Simple Chat Client & Server

Assigned: Fri 10 JAN 2025
Due: Sun 02 FEB 2025

Instructions:

- The assignment is to be uploaded to the course repository (GitHub) by the due date, which is scheduled for 11:59pm ET that day since solutions will be distributed soon after.
- We expect that you will study with friends and often work out problem solutions together, but *you must write up your own solutions, in your own words*. **Cheating will not be tolerated.** Professors and TAs will be available to answer questions but will not do your homework for you. One of our course goals is to teach you how to think on your own and solve your own problems using your resources.
- We require that all homework submissions be neat and organized. **There will be point deductions if the submission is not neat** (is disordered, difficult to read, etc.).

Pre-Assignment Work: Installation of GitHub Connection Locally

Git is a distributed source control system that allows developers collaborate and share source code. All assignments will be submitted using git. Install and configure git on your machine.

1. If on a Windows machine, download, install, and configure a github client from any of the following sites. Be sure to install the command tools as well:
   a. https://git-scm.com/download/win (preferred)
   b. https://desktop.github.com
   c. https://github.com/blog/1127-github-for-windows
2. If on a Mac, git should already be installed. Verify that git is installed on your machine by typing git at a terminal window

Setting Up GitHub

GitHub will be used to keep track of all work. All code will be committed and pushed to a private GitHub repository at GitHub.khoury.northeastern.edu. The GitHub repository will allow TAs and instructors easy access to your work. When submitting an assignment, tag the latest commit with a particular tag for that assignment provided by the assignment. This will be used by the TAs and instructors to grade the correct version of your code.

1. Create/access your account on GitHub.khoury.northastern,edu. If you already have a GitHub account, feel free to use the same account
2. Create a repository called CS5700_username, where username is YOUR username on GitHub. For MY username it would be CS5700_sav.
3. Commit and push your code to GitHub. In this example, sav is my GitHub username, and the new repository is CS5700_sav. You would use YOUR GitHub username, and YOUR new repository name.

```
> cd ~ (home directory)
> mkdir CS5700
> cd CS5700
> git init
> git commit -m "first commit"
> git remote add github https://github.khoury.northeastern.edu/sav/CS5700_sav.git
> git push -u github master
```

4. Click on the Code tab and verify that the source code is there
5. Invite the TAs (Duy Tran and Adit Shah) and the instructor (sav) as collaborators of the repository. This will allow them to help you debug your code if you run into trouble. It will also allow them to keep track of your progress. Go into Settings, then Collaborators. Search for the instructors and TAs by their username, and then click on Add collaborator.
6. Once you have created and connected to your private repository, connect to the class repository where the class files will be released. To do this, go to the CS5700 directory you created and complete the following steps:

> cd ~/CS5700 (home directory)
> git clone https://github.khoury.northeastern.edu/sav/CS5700_ClassRepo.git

Verify that you have received files from the class repository. In the future, to remain current with released files, simply issue a >git pull command inside the CS5700_ClassRepo directory and you will receive the most recent addition of files.

Problem 1 [100 points]: Simple Chat Service called "Speak"

Using the HW1 files in the ClassRepo, create a chat service called "speak" that allows the client to chat with a server user. Write a C client program called "speak" and a C server program called "speakd". These programs should communicate over a network using the socket interface to TCP on the Khoury Linux Server, but you can feel free to test your service on your local machine as you develop. You should use the sample client and server programs provided in the HW1 ClassRepo so that they can be used for a simple chat application.

Assume that the old-fashioned chat protocol is followed. The server is started up waiting for a client and the client gets to write first. The two programs alternate writing, with the current writer getting to write until control is relinquished. A "x" on a line by itself means that the writer is finished with writing and the writing control should be shifted to the other side (half-duplex communication). A "xx" on a line by itself by either side means that the chat is over and the connection should be terminated. You can compare strings with `strcmp` (see the `man` page for further information).

The essential part of the existing code that needs to change is the loop designs within the `client.c` and `server.c` files.

The message to be transferred between client and server can be formatted in any reasonably efficient manner you find convenient. The server must always send something back as a reply to any message received from the client. If the server detects an error from a system call, it should both print that error to its local `syserr` stream and send the error back to the client who will print the error on the client's local screen. The error message must contain the name of the file causing the error, and the text supplied by the stream routine `strerror()` on the server's host. (Since the error is detected by the server, only that server can reliably convert the `errno` value to an explanatory text message).

Grading
This assignment will have certain weighting associated with it. Your attempt to complete aspects will place you in line for greater reward at increased risk. To earn a grade of:

B: complete the assignment on the sample code simply as stated. Working code will solidify the B grade.

A: modify the code to make the user interface more elegant (whatever *you* deem that to mean) or use better program design than what has been presented to you already. **Include a README file with your code detailing the additional features included to qualify for the A grade.** Working code will solidify the A grade.

Submission

You must submit a "makefile" to compile and link your programs starting from the source files only. DO NOT MAIL OBJECT CODE. Please call your client's main program "speak" and your server's main program "speakd".

You should be able to run either program on the Khoury Linux Server.

All files should be placed into a folder called HW1 and included in your GitHub private repository.