CS 0449 – Sockets Lab

Introduction

To create a simple server, we need to remember all of the functions we discussed in class and put them in the proper order to get your program to accept connections. Connect to thot.cs.pitt.edu, cd to your /u/SysLab/USERNAME directory, and use your favorite text editor to make turing.c
We have a few steps to make a server using sockets:

1. Add the necessary header files

   ```
   #include <string.h>
   #include <sys/types.h>
   #include <sys/socket.h>
   #include <netinet/in.h>
   #include <arpa/inet.h>
   #include <stdio.h>
   ```

2. In the main function, create an integer called sfd to hold your socket descriptor and an instance of struct sockaddr_in called addr.

3. Initialize your socket:

   ```
   sfd = socket(PF_INET, SOCK_STREAM, 0);
   ```

   You should check this return value for -1, and if it is, return with an error.

4. Set up your structure. **IMPORTANT!** In order for everyone to do this we need to be using different port numbers. You can find the list of port numbers on my website (http://people.cs.pitt.edu/~aus/cs449/Ports.pdf)

   ```
   memset(addr.sin_zero, 0, sizeof(addr));
   addr.sin_family = AF_INET;
   addr.sin_port = htons(MYPORT);
   addr.sin_addr.s_addr = INADDR_ANY; //automatically find IP
   ```

5. Call bind to set up your port. Remember to check the return value against -1 and quit with an error message if it is. Bind sometimes fails if you try to run your program twice in a row too quickly, so this is an important one to check.

   ```
   bind(sfd, (struct sockaddr *)&addr, sizeof(addr));
   ```

6. Now call listen to setup the port as a server port, again checking for -1 on error.

   ```
   listen(sfd, 10);
   ```

7. Now you can wait for a connection using accept, which returns a file descriptor for the new connection:

   ```
   int connfd = accept(sfd, NULL, NULL);
   ```

8. The line after this (assuming no error… you did check right?) means you have a connection, so let's send a message:

   ```
   char buffer[1024];
   strcpy(buffer, "Hello there!");
   send(connfd, buffer, strlen(buffer), 0);
   ```

9. And let's close the sockets:

   ```
   close(connfd);
   close(sfd);
   ```

Testing

Compile and run the program you've created. Open an additional ssh window and connect to thot. At the terminal type:

```
telnet localhost PORT
```

where PORT is your personal port number.

You should see the output of your program, and the telnet connection close.

Have some fun

We named the file turing.c for a reason. The Turing Test is a test for Artificial Intelligence that supposes a convincing AI program would be indistinguishable from a person if you talked over a teletype, which telnet simulates. Make your server hold a conversation, and try to respond appropriately. For instance, if the person types “hi” (which you can read from the socket with `recv()`) respond back with a greeting. You don’t have to go overboard, but have some fun with it.

Show the TA when you are finished.