CPE 464 Lab – Socket API and using poll()

Due:

- Lab worksheet (pdf) – see canvas for due date.
- The lab programming assignment is in a different document – **do this worksheet first**.

Name: Rockwood Frank

Each student must type up their own answers. While you should work as a group, you are required to type up your worksheet and turn in a copy of your work.

I. If you worked in a group, please include your group members' names.

I. Socket API (Use my sockets lecture and Google to find answers to these questions)

This assumes you have watched the two videos on Canvas.

1. Write the C language command to create a TCP socket for IPv4/IPv6 family?
   socket(AF_INET6, SOCK_STREAM, 0)

2. Regarding the bind() function
   a. What information is needed to **name** a socket?
      Protocol family, IP Address, Port number
   b. Why do you only need to call bind() on the server? (so why do you need to call it on the server but not on the client)
      On the client, a program can implicitly get a port number since the kernel will assign one to you, so you don't need to bind your socket to a specific port number.
   c. How does the client get the server's port number?
      These are well known or reserved, or given at run time
   d. If we do not need to call bind() on the client, how does the client get assigned a port number? (this question is NOT about the server's port number but about the client getting its own port number)
      The kernel assigns a port to you
   e. How does the server get the client's port number?
      The information is given upon the return of an accept() call.

3. Regarding the accept() function
   a. When will the accept() function return (remember it's a blocking function)?
      It will return when a client connects to it.
   b. What does the accept() function return (besides -1 on error)?
      A new socket number for each accepted client

4. When using TCP on the **server** to communicate with clients there are at least two different sockets involved on the **server**. Explain:
   TCP needs two sockets, a welcome socket and a connection socket. The welcome socket makes initial connections with a client, and once that connection has been made, the other socket created so that that client can send data while other clients connect.
5. Regarding the poll() function (Google man poll):
   a. What does the poll() function do?
      The poll function waits for a file descriptor to be ready(have data to be processed) for the program. It blocks until it receives one of its specified events.
   b. Name/explain the parameters passed into the poll() function.
      Pollfd fds – a struct of file descriptors to check and which events to find
      Nfds: the number of file descriptors
      Timeout: the amount of time(in ms) to block for
   c. Regarding the timeout parameter passed to the poll() function
      i. What value will cause the function to block indefinitely?
         A negative value
      ii. What value will cause the function to not block but just look at the socket(s) and return immediately?
         A value of zero
      iii. How can the function be made to block for 3.5 seconds?
         Pass 3500

6. How does the server (or client) know that the client (or server) has terminated (e.g. ^C, segfault) when using the recv() call?
   The function will return zero or -1 and will set errno to ECONNRESET

7. Explain why we need to use the MSG_WAITALL flag on the recv() function call in program #2.
   The recv function just returns a stream of data, so we make the first two bytes of our PDU the PDU length, and then once we have that, we need to wait until all of that data is available, to prevent data from being mismatched.

8. Regarding the code provided with the lab (it is the same code provided with programming assignment #2). The purpose of these questions is to help you understand the code I have provided.

   These questions all concern the file: **networks.c**

   a. Write the line of code (from network.c) that creates the server_socket:

      mainServerSocket= socket(AF_INET6, SOCK_STREAM, 0);

   b. What are the line numbers for the code that is used to name the server's socket?

      44-46

   c. What is the effect of the following:

      - What does the In6addr_any do (so what does bind() do)?

        (e.g. serverAddress.sin6_addr = in6addr_any;)

        It's a wildcard that tells the system to use that system's IP address for the socket's IP address.

      - If I use a port of 0 in my call to bind(), what does bind() do?

e.g.

```
serverPort = 0;
serverAddress.sin6_port= htons(serverPort);
```

The kernel will assign a port number to you.

II. Regarding TCP:

There are several function calls that must be made on a client and server in order to utilize sockets for communications (all in networks.c except send()/recv() and close() which are in myClient.c and myServer.c).

1. In the table below, list the function names (see socket video slide 4) in the order which are needed to setup, use and teardown a connection using Stream Sockets (get this list of functions from the video slides on sockets).
2. In the table below, connect the functions that are part of the connection oriented part of TCP using arrows.
3. Looking at the network.c file provided with this lab put the line number of the call to that function in the table. (Don't worry about line numbers for send()/recv() and close() since these are called in myClient.c and myServer.c.)

Client Line #	**Functions called on the Client**	**Functions called on the Server**	Server Line #
		socket()	36
		Bind()	49
106	socket()	getsockname()	56
118	getHostByName()	listen()	62
123	connect()	accept()	82
	Send()->	->recv()	
	Recv()<-	<-send()	
	Close()	close()	