CS 125
Sockets
See Calendar for Dates

January 31, 2014

6:58 PM

Introduction

The purpose of this assignment is implement three network functions, DNS access, a socket version of echo, and a threaded socket version of echo.. The lab can (almost) be accomplished by slight modification and **documentation** of the code found in the CS 105 text and presented in CS 105 lecture.

Logistics

You MUST work in a group of at least two people in solving the problems for this assignment. The only "hand-in" will be electronic, but you will demonstrate your code in operation. Any clarifications and revisions to the assignment will be posted on the course Web page. **We strongly recommend that you and your partner brainstorm before coding. If you are having trouble attacking this problem, look at Mike's Guide**

Handout Instructions

Basically, you are to implement **DNS access** and **echo** using sockets and I/O.

DNS Access Instructions

You are to write a DNS access function, **dnsaccess.c**, that takes input from the command line (either an IP Address or a fully qualified DNS name) and prints out **all** the related IP addresses and names. You are creating your own version of *NSlookup*. To implement this

function you are to use the new DNS system routines, i.e., *getaddrinfo*. **dnsaccess.c**, is a standalone program, the guts of which you will need in implementing **echo**.

Socket Instructions

You are to follow the book example with the following additions/clarifications:

- Do not download any code from the CMU website or anywhere else, i.e., enter the code yourselves.
- Make the Server name or IP Address an **input parameter** as needed to your Client.
- Modify your **dnsaccess** to get only the single IP address that you will need for the connection to the Server.
- Make the Server port id an **input parameter** to your Client. But, when running use one partner's UNIX uid as the port number for your Server (check out the **id** command). This should avoid conflicts with other services, and with other users.
- Use IPv4.
- Run your Server on `Wilkes` and your Client on `Knuth` or some lab machine.
- **Do not** use any of the encapsulated functions provided in the CS105 text, e.g., `open_clientfd`. Rather, write your own versions, making changes to use commands other than those of RIO, i.e., use standard C or Unix IO libraries.
- Use calls to DNS via a modified version of your **dnsaccess** program. Again, your **dnsaccess** should use the 'new' versions of the dns functions, e.g., **getnameinfo**, **getaddrinfo**, etc. to get the Server IP address. Make sure to **COMMENT WELL** the various fields of the DNS data structure.
- Catch any and all errors that might be generated by any of your library calls. The extreme of error handling is to recognize an error and then check the details of the individual error code. You only need to recognize the error, and quit appropriately.
- Have the Server and the Client use print statements to indicate what is happening. See the text for examples. For example, each Server or Client print of the echo exchange should begin with: **server: received and echoed XX bytes from Client IPAddress** or **client: sent XX bytes, "the string..."** .
  client: received XX bytes, "the string..." .
  When completed have the Server and Client print out the total number of bytes exchanged.
- Document your code very well, i.e., you should mention the man pages and library functions that your found particularly helpful.

Threaded Sockets Instructions

In this version your echo server will be able to handle many simultaneous requests from echo clients, i.e., a thread is created to handle each new echo request.

- Start with the code you did above.
- Thread the server so that it can handle multiple simultaneous echo requests.
- Again use your DNS access function.
- Again, for the Client, make the Threaded Echo Server port number a **parameter** as needed.
- Run your server on `Wilkes` and your clients on other machines. You might write a script to launch a number of echo requests on a single machine. (but we want the servers and the clients on different machines so that the network comes into play).
- Again, have the server and the client use print statements to indicate what is happening. In particular each server print statement should begin with:
  Thread Server: Client IP/ClientPortNumber – theechotext
  or **Process Server: Client IP/ClientPortNumber – theechotext**
  In particular each client print statement should begin with:
  Client: sent – thesenttext.. .
  Client: echo – theechotext.. .
  When completed have the server and client print out the total number of bytes exchanged, again the server is to identify the port number on the client.

What to Turn-in

See the Lab web page to determine the assignment number.

- dnsaccess.c
- echoserver.c
- echoclient.c
- Techoserver.c (threaded echo server)
- `script` session for all the Clients and the Server of at least one interaction as a Threaded echo server. Submit these as a single text file,
- Make sure to include team member names in the comments.
- Be prepared to demonstrate your program to one of the Grutors or mike.

IMPORTANT NOTE

Threads have a problem related to a unique copy of the Socket information. If you understand the problem, be sure to include comments in your code heading describing the problem and the appropriate solution. In all our tests this problem will not show up because we are not fast enough with a request for a new connection.