Distributed Computing

CSCI 5210 Assignment 3 Fall 2015

Building Client-Server Programs Using Sockets

Date Assigned: Wed. 28^{th} Oct. Due Date: Wed. 11^{th} Nov.

Objective: The goal of this homework is to get you familiar with Application Programming Interface for Berkeley Sockets. In order to achieve this goal you will implement a simple client-server application. You will develop a connection oriented client and a connection oriented concurrent server using Internet Prtocol Version 4.

Client-Server Application: As usual, our actual application will be rather simple so that you can focus your energy on learning how to use sockets. The programs should do the following:

1. The client reads a line from its standard input and writes the line to the server. The line is intended to be the name of a State in USA.

2. The server reads a line from its network input. The server then consults a datafile called "capitalinfo" and determines the capital of the State that it got from the client and sends that line back to the client. If the input is not a State, it sends "Not a State" back to the client. So, we shall call this the Capital Server.

3. The client reads a line from its network input and prints it on its standard output.

The client should be able to process several states, one after the other, in a single invocation. You may assume that the state names entered will follow the standard convention regarding capitalization and spaces. A copy of the "capitalinfo" file will be emailed to you.

Some Choices: Make sure to implement the client as a connection oriented client. Similary, make sure to implement the server as a connection oriented server. Further make sure to implement the server as a concurrent server rather than an iterative server. Make sure to use Internet Protocol Version 4 (PF_INET) as the protocol family and to use stream type of sockets (SOCK_STREAM).

Basic Approach: The basic approach in building a connection oriented client-server application is diagrammed in a separate page of this handout. Use this approach in building your client program and the server program. The term concurrent server means that the server process itself should not really do the service. Once it gets a request, it should immediately fork off a new process (or create a new thread) which should provide the service. This way the server can handle the next request it gets without really waiting for the first request to be processed completely. Make sure to test that your server can handle multiple clients.

The system calls that you will need on the server side are `socket`, `bind`, `listen`, `accept`, `read`, `write` and `close`. The system calls that you will need on the client side are `socket`, `connect`, `write`, `read` and `close`. Make sure to read the man pages of these system calls to get a full understanding of them so that you can use them appropriately.

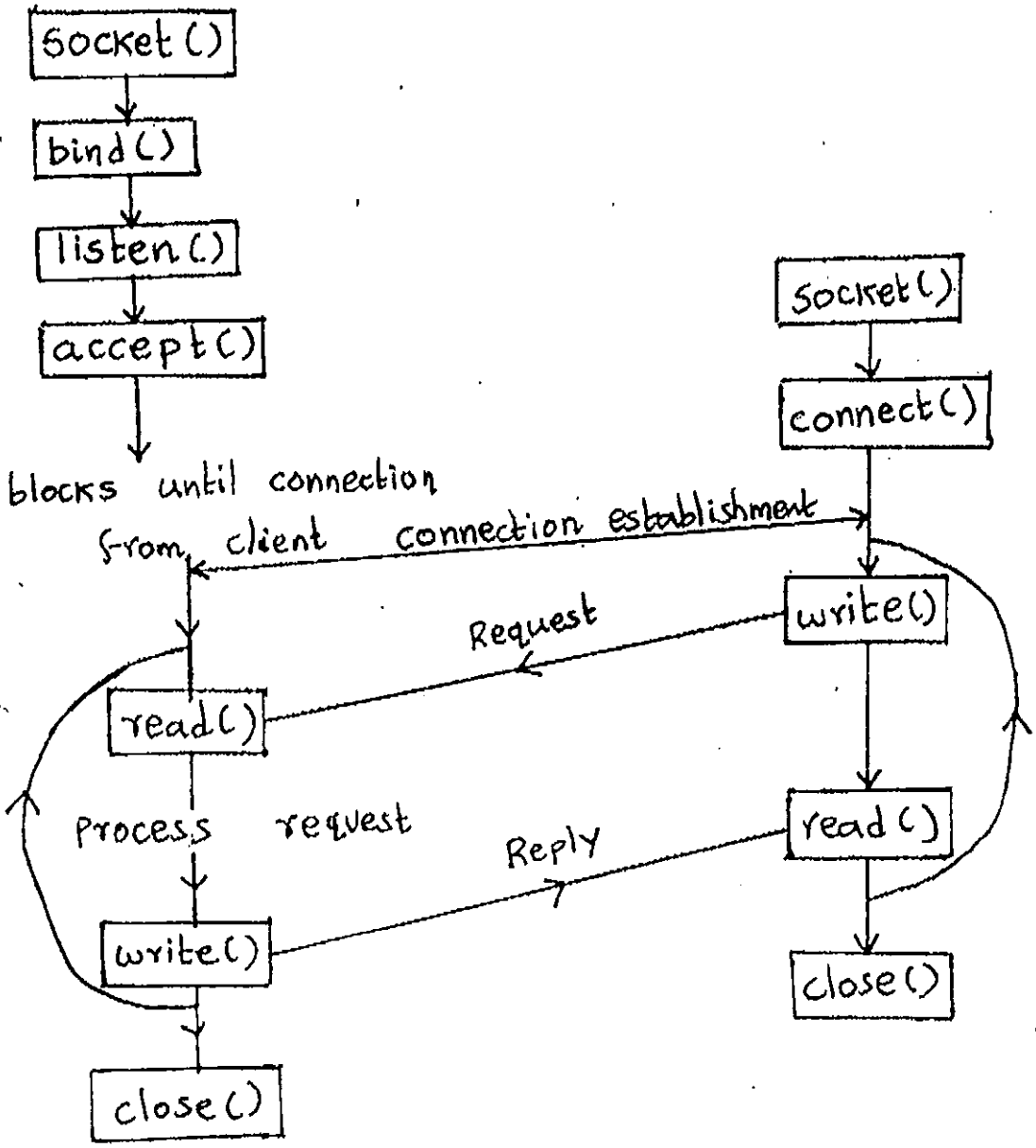
General Instructions: Use the thin clients in Room No. 208 or the PCs in Room No. 207 or your own computer to get onto `xlogin.cs.ecu.edu` and work on your programming assignments. The `xlogin` server runs SUSE LINUX. Create a subdirectory called *assign3* in your csci5210 directory. Use that subdirectory to store all the files concerning this assignment and nothing else. You need to follow these general guidelines for all your future assignments as well. Name the two programs *cserver.c* and *cclient.c* If you are not using makefile, please include the name of the compiler you are using and any special options needed as comments to your source code. Additionally, you can also handin a *readme* file describing the assumptions, limitations and usage of your programs. Feel free to make any reasonable assumption and document it in the *readme* file. Get started early.

Further Instructions: Use the (virtual) machine *pogues.cs.ecu.edu* as the server and the virtual machine *xlogin.cs.ecu.edu* as the client for this assignment. The IP address of *pogues* is 150.216.150.42. Ideally, you should write the client in such a way that it takes one command line parameter, viz., the name of the server machine and then use that name to figure out the corresponding IP address within your program. However, to begin with you can hardcode the IP address of the server into the socket address structure used in the client's program. Make sure to test that your server can handle requests from multiple clients.

What to handin? Turn in all the files using the *submit* utility. To submit your program type the following command

```
~gopal/cs5210/bin/submit 3  cserver.c cclient.c
```

The first component invokes the `submit` program. The second component is the assignment number. The last component is the list of files that you wish to submit. Make sure to submit the complete set of files that you wish to submit (including "makefile" if any and "readme" file if any). Your programs will be compiled and tested and then the scores will be sent to you by email.


socket()
bind()
listen()
accept()
blocks until connection from client
socket()
connect()
connection establishment
write()
Request
read()
Process request
read()
Reply
write()
close()
close()