CS 471 Homework 1

Due Friday Friday 12
Homework 1

Implement a client/server application that uses TCP sockets. The client sends one or more messages to the server. Each message contains one or more integers greater than one. The server responds to a client message containing n integers by sending a message containing n integers. The integers in the reply are the largest prime numbers less than or equal to each integer sent by the client. For example, when the client sends a message containing the integers 18, 7, 15, 28 the server should respond with a message containing the integers 17, 7, 13, 23.

The client can send multiple messages to the server but after sending a message to the server the client waits for a response from the server before it sends another message to the server. You must develop your own protocol for the message formats and a way for the client to tell the server it (the client) is finished.
Homework 1

The server must be able to serve multiple clients simultaneously. The server implementation must use pthreads. You can assume the server runs an infinite loop in which it waits for new connection requests like I have done in the example programs.

To test your program assume the client expects three command line arguments: the IP address of the server, the port number of the server and the name of a text file. The IP address will use the dotted quad notation. Each line in the text file will contain between 1 and 100 integers. The integers will be separated by a single blank. The integers in one line should be sent as a single message to the server. The client should print to standard output the values returned by the server.

The server expects one command line argument: the port number on which it will accept connections.
First develop the program so it works on a single machine (using 127.0.0.1). After the program is working on a single machine, ssh onto multiple machines and run multiple clients.
Homework 1 Submission

You will demonstrate your program to me in Wing 16. When you demonstrate your program I will have you run three clients and a server. Nothing in your program should be dependent on the fact that I will test it with three clients.
sockfd = socket(AF_INET, SOCK_STREAM, 0);

memset(&server, 0, sizeof(struct sockaddr_in));
server.sin_family = AF_INET;
server.sin_addr.s_addr = inet_addr(argv[1]);
server.sin_port = htons(40000);

connect(sockfd, (struct sockaddr *) &server, sizeof(struct sockaddr_in));
nums[0] = atoi(argv[2]);
nums[1] = atoi(argv[3]);
nums[0] = htonl(nums[0]);
nums[1] = htonl(nums[1]);
send(sockfd, nums, 2*sizeof(int),0);
int len = recv(sockfd, &result, sizeof(int), 0);
printf("%d\n", ntohl(result));
addServer (Partial)

sockfd1 = socket(AF_INET, SOCK_STREAM, 0);

memset(&server, 0, sizeof(struct sockaddr_in));
server.sin_family = AF_INET;
server.sin_addr.s_addr = INADDR_ANY;
server.sin_port = htons(40000);

bind(sockfd1, (struct sockaddr *) &server, sizeof(struct sockaddr_in));

listen(sockfd1, 5);

sockfd2 = accept(sockfd1, (struct sockaddr *) &client, &clen);
len = recv(sockfd2, nums, 2*sizeof(int), 0);
result = ntohl(nums[0])+ntohl(nums[1]);
result = htonl(result);
send(sockfd2, &result, sizeof(int), 0);
numbers = fopen(argv[2], "r");

int ret = connect(sockfd, (struct sockaddr *) &server, sizeof(struct sockaddr_in));

while (fscanf(numbers, "%d %d", &nums[0], &nums[1]) != EOF) {
    nums[0] = htonl(nums[0]);
    nums[1] = htonl(nums[1]);
    send(sockfd, nums, 2*sizeof(int), 0);
    int len = recv(sockfd, &result, sizeof(int), 0);
    printf("%d\n", ntohl(result));
}

close(sockfd);
addServer2 (Partial)

```c
pthread_t tid;

sockfd1 = socket(AF_INET, SOCK_STREAM, 0);

memset(&server, 0, sizeof(struct sockaddr_in));
server.sin_family = AF_INET;
server.sin_addr.s_addr = INADDR_ANY;
server.sin_port = htons(40000);

bind(sockfd1, (struct sockaddr *) &server, sizeof(struct sockaddr_in));

listen(sockfd1, 5);

while (1) {
    sockfd2 = accept(sockfd1, (struct sockaddr *) &client, &clen);
    pthread_create(&tid, NULL, add, (void *) sockfd2);
}
```
addServer2 (Partial)

```c
void *add(void *s) {
    int nums[2];
    int result;
    int sockfd = (int) s;

    while ( recv(sockfd, nums, 2*sizeof(int), 0) != 0 ) {
        result = htonl(nums[0])+ntohl(nums[1]);
        result = htonl(result);
        send(sockfd, &result, sizeof(int), 0);
    }
    close(sockfd);
    return NULL;
}
```