Networked Applications NWEN 243

Lab Exercise 2 (2 points) – Introduction to TCP

Objectives

- Experience using TCP
- Next week, you will extend this and create a TCP server

Requirements

- This lab an individual lab, written in C.
- We will be writing programs that you execute from the shell command line.
- You must demonstrate your work to your lab demonstrator to gain credit.

Preliminaries

TCP – or the transmission control protocol, is a reliable byte ordered transport layer service. Delivery and order are guaranteed.

To use TCP you will need to use the Socket API. The socket API is modeled on the file system, so uses READ and WRITE to access the network.

The Exercise

Your tutor will run a TCP server that capitalizes a sentence (the SHOUTING server). They will advise you of the port number and machine.

Your task is to complete a client program that will

1. connect to the server,
2. pass in a string (obtained from the command line),
3. read the server's response, and
4. write the response to the screen.

Resources

- There is a C skeleton, in the usual place on the course lab pages.

Useful stuff for C

You will need to use the following functions from the socketAPI.

int socket(int domain, int type, int protocol);

> socket() creates an endpoint for communication and returns a socket descriptor. The domain parameter specifies a communication domain; this selects the protocol family which will be used for communication. You should use AF_INET. For a TCP connection, the type should be

SOCK_STREAM, unlike last week when we used SOCK_DGRAM for UDP, the protocol should be 0.

*int connect(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen);*

The connect() system call connects the socket referred to by the socket descriptor to the address specified by serv_addr. The addrlen argument specifies the size of serv_addr. The server address structure is provided in the C skeleton.

*int read(int sockfd, void *buf, int count);*

read() attempts to read up to count bytes from file descriptor fd into the buffer starting at buf.

*int write(int sockfd, const void *buf, int count);*

write() writes up to count bytes to the file referenced by the file descriptor fd from the buffer starting at buf.

int close(int fd);

close() closes a file descriptor, so that it no longer refers to any file and may be reused.

Notes:

1. All above functions return a value < 0 to indicate an error. You need to check for this.
2. Make sure you don't overflow any buffers, limit the sizes correctly.

Hand in

Demo your work to your lab demonstrator.