461 Socket Programming Assignment

Due 02/11 (before the precept)

This assignment introduces you to basic socket programming as discussed in Precept 1. The goal is to get you comfortable with (a) Unix socket programming and (b) simple client-server interaction.

You are required to turn in code which can be typed or handwritten. The code is not required to compile. Emphasis is on the correctness of the code (buffer management, handling return values), system calls and their arguments.

Grading: This assignment will count towards your participation grade.

We want you to write a server capable of receiving text messages from clients. The server should print these text messages on the standard output. From the server perspective, the message corresponds to the data received from a particular client during a communication session with that client. The server should be listening for text messages on a port known to the clients. It should handle the client connections sequentially and accept connections from multiple clients. After servicing one client to completion, it should proceed to the next. If multiple clients try to simultaneously send text messages to the server, the server should handle them one at a time (in any order). Serving one client at a time is enough.

We also want you to write a client. The client should read from a file, transmit the message and exit.

You can assume that the client is run as "client <server> <port>" where <server> is the IP address of the server, and <port> is the port the server is listening on. The server is run as "server <port>". If the server cannot bind on a port, print a message to standard error.

Make sure you handle the following correctly:

1. Buffer management: Assume that the file contents can be arbitrarily large (assume a typical size of 20KB), but the buffers you use to read/write to the file or socket must be small and of fixed size (e.g., 4096 bytes).

2. Handling return values: By default, sockets are blocking, and for this assignment we will use only blocking sockets. "Blocking" means that when we issue a socket call that cannot be done immediately (including not being able to read or write), our process is waits until it can perform the action. This includes the case when

(a) a socket's internal buffer is full and therefore, no data can be written, or
(b) a socket's buffer is empty, and no data is available to be read.

However, if there is some data available to be read or some can be written, the call will return the number of bytes read or written respectively. NOTE: This returned value can be less than specified in the length argument to the call or indicate an error. You must handle this.