

Client-Sever Programming Using TCP Sockets

INSTALLATION:

Create a new subdirectory containing CliSer.tar, then issue

```
> tar xvf CliSer.tar
> make
```

Four new binary executables, named tcpCli, tcpSer, udpCli and udpSer will be created in that same subdirectory.

EXECUTION:

This example uses the demonstration client program, named tcpCli, and a server program, named tcpSer. They communicate over a tcp stream socket on port number 6556. It assumes that you have two nodes on your network, named alpha and beta.

1. Start tcpSer on beta, as follows

```
> ./tcpSer
```

Note: You must type Ctrl-C to exit tcpSer if a "broken pipe" doesn't kill it first.

2. Then start tcpCli on alpha, as follows

```
> ./tcpCli beta
```

Note: It is important that you substitute your <hostname> for "beta" in order to tell tcpCli where the server program, tcpSer, is running. Type Ctrl-C to exit.

Program tcpCli is waiting for you to type a simple arithmetic expression, like

```
6 + 7
```

Program tcpCli sends this expression to program tcpSer via a tcp stream socket for processing. Program tcpSer "solves" the expression and sends it back to tcpCli for display on the client terminal:

```
6 + 7 = 13
```

Type another simple expression, or type Ctrl-C to exit. Programs udpCli and udpSer are identical, except they use UDP sockets instead of TCP to communicate.

There is not much error checking included in these sample programs. Try adding code in tcpCli to detect a null string input, sending a "quit" message to tcpSer, for example, so both client and server exit gracefully. Have fun!