

TCPEchoServer.c

From TCP/IP Sockets in C: Practical Guide for Programmers, Donohoo and Calvert.

```
#include <stdio.h>          /* for printf() and fprintf() */
#include <sys/socket.h>     /* for socket(), bind(), and connect() */
#include <arpa/inet.h>     /* for sockaddr_in and inet_ntoa() */
#include <stdlib.h>        /* for atoi() and exit() */
#include <string.h>        /* for memset() */
#include <unistd.h>        /* for close() */

#define MAXPENDING 5      /* Maximum outstanding connection requests */

void DieWithError(char *errorMessage); /* Error handling function */
void HandleTCPClient(int clntSocket); /* TCP client handling function */

int main(int argc, char *argv[])
{
    int servSock;          /* Socket descriptor for server */
    int clntSock;         /* Socket descriptor for client */
    struct sockaddr_in echoServAddr; /* Local address */
    struct sockaddr_in echoClntAddr; /* Client address */
    unsigned short echoServPort;    /* Server port */
    unsigned int clntLen;           /* Length of client address data structure */

    if (argc != 2) /* Test for correct number of arguments */
    {
        fprintf(stderr, "Usage:  %s <Server Port>\n", argv[0]);
        exit(1);
    }

    echoServPort = atoi(argv[1]); /* First arg:  local port */

    /* Create socket for incoming connections */
    if ((servSock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0)
        DieWithError("socket() failed");

    /* Construct local address structure */
    memset(&echoServAddr, 0, sizeof(echoServAddr)); /* Zero out structure */
    echoServAddr.sin_family = AF_INET; /* Internet address family */
    echoServAddr.sin_addr.s_addr = htonl(INADDR_ANY); /* Any incoming interface */
    echoServAddr.sin_port = htons(echoServPort); /* Local port */

    /* Bind to the local address */
    if (bind(servSock, (struct sockaddr *) &echoServAddr, sizeof(echoServAddr)) < 0)
        DieWithError("bind() failed");

    /* Mark the socket so it will listen for incoming connections */
    if (listen(servSock, MAXPENDING) < 0)
        DieWithError("listen() failed");

    for (;;) /* Run forever */
    {
        /* Set the size of the in-out parameter */
        clntLen = sizeof(echoClntAddr);

        /* Wait for a client to connect */
```

```

    if ((clntSock = accept(servSock, (struct sockaddr *) &echoClntAddr,
        &clntLen)) < 0)
        DieWithError("accept() failed");

    /* clntSock is connected to a client! */

    printf("Handling client %s\n", inet_ntoa(echoClntAddr.sin_addr));

    HandleTCPClient(clntSock);
}
/* NOT REACHED */
}

```

HandleTCPClient.c

From TCP/IP Sockets in C: Practical Guide for Programmers, Donohoo and Calvert.

```

#include <stdio.h>          /* for printf() and fprintf() */
#include <sys/socket.h>    /* for recv() and send() */
#include <unistd.h>        /* for close() */

#define RCVBUFSIZE 32     /* Size of receive buffer */

void DieWithError(char *errorMessage); /* Error handling function */

void HandleTCPClient(int clntSocket)
{
    char echoBuffer[RCVBUFSIZE];      /* Buffer for echo string */
    int recvMsgSize;                  /* Size of received message */

    /* Receive message from client */
    if ((recvMsgSize = recv(clntSocket, echoBuffer, RCVBUFSIZE, 0)) < 0)
        DieWithError("recv() failed");

    /* Send received string and receive again until end of transmission */
    while (recvMsgSize > 0)           /* zero indicates end of transmission */
    {
        /* Echo message back to client */
        if (send(clntSocket, echoBuffer, recvMsgSize, 0) != recvMsgSize)
            DieWithError("send() failed");

        /* See if there is more data to receive */
        if ((recvMsgSize = recv(clntSocket, echoBuffer, RCVBUFSIZE, 0)) < 0)
            DieWithError("recv() failed");
    }

    close(clntSocket); /* Close client socket */
}

```