
```c
/*********************************************
 *
 *                  server.c
 *
 *  A data gram based echo server.
 *********************************************/

#include <arpa/inet.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char **argv) {
    int sock;
    int arg1, arg2;
    int i;
    int rc;
    struct sockaddr address;
    socklen_t addrLength = sizeof(address);
    struct addrinfo hints;
    struct addrinfo *addr;
    char *message;

    /*
     *  set the hints structure to zero
     */
    memset(&hints, 0, sizeof(hints));
```

```
    /*
     *  want a datagram, also address that will accept all
     *  connections on this host
     */
    hints.ai_socktype = SOCK_DGRAM;
    hints.ai_flags = AI_PASSIVE | AI_ADDRCONFIG;
    if((rc = getaddrinfo(NULL, "4321", &hints, &addr))) {
        printf("host name lookup failed: %s\n", gai_strerror(rc));
        exit(1);
    }

    /*
     *  use the first entry returned by getaddrinfo
     */
    sock = socket(addr->ai_family, addr->ai_socktype, addr->ai_protocol);
    if(sock < 0) {
        printf("Can't create socket\n");
        exit(1);
    }

    /*
     *  want to be able to reuse the address right after
     *  the socket is closed.  Otherwise must wait for 2 minutes
     */
    i = 1;
    setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &i, sizeof(i));

    rc = bind(sock, addr->ai_addr, addr->ai_addrlen);
    if(rc < 0) {
        printf("Can't bind socket\n");
        exit(1);
    }

    /*
     *  free results returned by getaddrinfo
     */
    freeaddrinfo(addr);
    /*
     *  loop receiving a string from client and echoing it back
     */
```

```
    while(1) {
        /*
         *  read message from client and respond
         */
        recvfrom(sock, (char*) &arg1, sizeof(arg1), 0, (struct sockaddr*)
&address,  &addrLength);
        recvfrom(sock, (char*) &arg2, sizeof(arg2), 0, (struct sockaddr*)
&address,  &addrLength);
        printf("Data from client: %d %d\n", arg1, arg2);
        arg1 = arg1+arg2;
        sendto(sock, (char *) &arg1, sizeof(arg1), 0,
                (const struct sockaddr*) &address, addrLength);

        // recvfrom(sock, (char*) &message, sizeof(message), 0, (struct
sockaddr*) &address,  &addrLength);
        // if(message != NULL) {
        //  printf("Message from client: %s\n", message);
        //  free(message);
        // } else {
        //  printf("Error receiving message from client\n");
        // }
    }
    close(sock);
    exit(0);
}

/*****************************************************
 *
 *                          client.c
 *
 *  Client program to test the data gram based echo server.
 *****************************************************/

#include <arpa/inet.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
```

```c
#include <unistd.h>

int main(int argc, char **argv) {
    struct addrinfo hints;
    struct addrinfo *addr;
    struct sockaddr_in *addrinfo;
    int rc;
    int sock;
    int result;
    int arg1, arg2;

    if (argc != 3) {
        printf("Usage: client num1 num2\n");
        exit(1);
    }

    arg1 = atoi(argv[1]);
    arg2 = atoi(argv[2]);

    /*
     *  clear the hints structure to zero
     */
    memset(&hints, 0, sizeof(hints));

    /*
     *  want a data gram on a compatible interface
          */
    hints.ai_socktype = SOCK_DGRAM;
    hints.ai_flags = AI_ADDRCONFIG;
    /*
     *  localhost is the name of the current computer
     */
    rc = getaddrinfo("localhost", NULL, &hints, &addr);
    if(rc != 0) {
        printf("Host name lookup failed: %s\n", gai_strerror(rc));
        exit(1);
    }

    /*
     *  use the first result from getaddrinfo
```

```
     */
    addrinfo = (struct sockaddr_in *) addr->ai_addr;

    sock = socket(addrinfo->sin_family, addr->ai_socktype,
addr->ai_protocol);
    if(sock < 0) {
        printf("Can't create socket\n");
        exit(1);
    }

    /*
     *  specify the port number
     */
    addrinfo->sin_port = htons(4321);

    /*
     *  free the results returned by get addrinfo
     */
    freeaddrinfo(addr);

    /*
     *  loop reading a line from the user and sending it
     *  to the echo server, print the line when if comes back
     */
    sendto(sock, (char*) &arg1, sizeof(arg1), 0,
            (const struct sockaddr*) addrinfo, addr->ai_addrlen);
    sendto(sock, (char*) &arg2, sizeof(arg2), 0,
            (const struct sockaddr*) addrinfo, addr->ai_addrlen);
    recvfrom(sock, (char*) &result, sizeof(result), 0, NULL, NULL);
    printf("The sum of %d and %d is %d\n", arg1, arg2, result);

    // char *ty = "Thank You";
    // sendto(sock, (char*) &ty, sizeof(ty), 0,
    //      (const struct sockaddr*) addrinfo, addr->ai_addrlen);

    close(sock);
    exit(0);
}
```