

```

//===== file = udpServer.c =====
//= A message "server" program to demonstrate sockets programming
=
//=====
==
//= Notes:
=
//= 1) This program conditionally compiles for Winsock and BSD sockets.
=
//= Set the initial #define to WIN or BSD as appropriate.
=
//= 2) This program serves a message to program udpClient running on
=
//= another host.
=
//-----
--
//= Example execution: (udpServer and udpClient running on host 127.0.0.1)
=
//= Waiting for recvfrom() to complete...
=
//= IP address of client = Computernetworkassignment
//= Received from client: Test message from CLIENT to SERVER
=
//-----
--
//= Build: bcc32 udpServer.c or cl udpServer.c wsock32.lib for Winsock
=
//= gcc udpServer.c -lsocket -lnsl for BSD
=
//-----
--
//= Execute: udpServer
=
//-----
--
//=====
==
#define WIN // WIN for Winsock and BSD for BSD sockets

//----- Include files -----
-
#include <stdio.h> // Needed for printf()
#include <string.h> // Needed for memcpy() and strcpy()
#include <stdlib.h> // Needed for exit()
#ifdef WIN
#include <windows.h> // Needed for all Winsock
#endif
#ifdef BSD
#include <sys/types.h> // Needed for sockets
#include <netinet/in.h> // Needed for sockets
#include <sys/socket.h> // Needed for sockets
#include <arpa/inet.h> // Needed for sockets
#include <fcntl.h> // Needed for sockets
#include <netdb.h> // Needed for sockets
#endif

```

```

//----- Defines -----
-
#define PORT_NUM 1050 // Arbitrary port number for the server

//===== Main program
=====
int main()
{
#ifdef WIN
WORD wVersionRequested = MAKEWORD(1,1); // for WSA functions
WSADATA wsaData; // for WSA functions
#endif
int server_s; // Server socket descriptor
struct sockaddr_in server_addr; // Server Internet address
struct sockaddr_in client_addr; // Client Internet address
struct in_addr client_ip_addr; // Client IP address
int addr_len; // Internet address length
char out_buf[4096]; // Output buffer for data
char in_buf[4096]; // Input buffer for data
int retcode; // Return code

#ifdef WIN
// This initializes winsock
WSAStartup(wVersionRequested, &wsaData);
#endif

// >>> Step #1 <<<
// Create a socket
// - AF_INET is Address Family Internet and SOCK_DGRAM is datagram
server_s = socket(AF_INET, SOCK_DGRAM, 0);
if (server_s < 0)
{
printf("*** ERROR - socket() failed \n");
exit(-1);
}

// >>> Step #2 <<<
// Fill-in my socket's address information
server_addr.sin_family = AF_INET; // Address family to use
server_addr.sin_port = htons(PORT_NUM); // Port number to use
server_addr.sin_addr.s_addr = htonl(INADDR_ANY); // Listen on any IP
address
retcode = bind(server_s, (struct sockaddr *)&server_addr,
sizeof(server_addr));
if (retcode < 0)
{
printf("*** ERROR - bind() failed \n");
exit(-1);
}

// >>> Step #3 <<<
// Wait to receive a message from client
printf("Waiting for recvfrom() to complete... \n");
addr_len = sizeof(client_addr);
retcode = recvfrom(server_s, in_buf, sizeof(in_buf), 0,
(struct sockaddr *)&client_addr, &addr_len);
if (retcode < 0)

```

```

{
    printf("*** ERROR - recvfrom() failed \n");
    exit(-1);
}

// Copy the four-byte client IP address into an IP address structure
memcpy(&client_ip_addr, &client_addr.sin_addr.s_addr, 4);

// Print an informational message of IP address and port of the client
printf("IP address of client = %s port = %d \n",
inet_ntoa(client_ip_addr),
    ntohs(client_addr.sin_port));

// Output the received message
printf("Received from client: %s \n", in_buf);

// >>> Step #4 <<<
// Send to the client using the server socket
strcpy(out_buf, "This is a reply message from SERVER to CLIENT");
retcode = sendto(server_s, out_buf, (strlen(out_buf) + 1), 0,
    (struct sockaddr *)&client_addr, sizeof(client_addr));
if (retcode < 0)
{
    printf("*** ERROR - sendto() failed \n");
    exit(-1);
}

// >>> Step #5 <<<
// Close all open sockets
#ifdef WIN
retcode = closesocket(server_s);
if (retcode < 0)
{
    printf("*** ERROR - closesocket() failed \n");
    exit(-1);
}
#endif
#ifdef BSD
retcode = close(server_s);
if (retcode < 0)
{
    printf("*** ERROR - close() failed \n");
    exit(-1);
}
#endif

#ifdef WIN
// This cleans-up winsock
WSACleanup();
#endif

// Return zero and terminate
return(0);
}

```