```
// ***** udpclient.cpp *****

//*************************************************************************
//
// Computer Science 4/5313 Computer Networks
//
// Spring  2016
//
// Instructor:  Hung-Chi Su
//
// Assignment # n
//
// Programmer:  your name
//
// Due Date:    day-of-week, month day, year
//
// Description: This is a UDP socket program that illustrates how a client
//              talks to a UDP server (udpserver.cpp) and sends a message
//              to the server and receives a response.
//
// Editor/Platform:  vi/Linux
//
//
//
//  Input:      none / let user input some data
//
//  Output:     The client will display the message replied by the server
//              The server will display the message sent by this client
//
//  Compile:    g++ -o udpclient1 udpclient.cpp
//
//  Command:    (After server is running)
//              ./udpclient1 [<port#>  [<IP>]]
//
//  Note:       Remember to Change SERVER_PORT_ID to be same as server's
//
//*************************************************************************

                        // header files

//#include <stdio.h>    //for printf(), ...
#include <stdlib.h>
#include <string.h>
//#include <sys/types.h>        //for data types
#include <sys/socket.h> //for socket(), connect(), ...
#include <unistd.h>     //for close()
#include <netinet/in.h> //for internet address family
#include <arpa/inet.h>  //for sockaddr_in, inet_addr()
#include <errno.h>      //for system error numbers
```

```
#include <iostream>     //for cin, cout...


// !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// change the last 4 digits of SERVER_PORT_ID to your last 4 digits of student ID
// !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#define SERVER_PORT_ID  11073

// #define SERV_HOST_ADDR "147.97.156.237"
#define SERV_HOST_ADDR "147.97.156.237"
#define MESSAGESIZE        80

using std::cout;
using std::cin;
using std::cerr;

main(int argc, char *argv[])
{
  int sockfd, retcode, nread, addrlen;
  int server_port_id = SERVER_PORT_ID;
  char serv_host_addr[20]=SERV_HOST_ADDR;
  struct sockaddr_in my_addr, server_addr;
  char msg[MESSAGESIZE];

  if (argc>1)  // server port (and server IP) is provided
  {
      server_port_id = atoi(argv[1]);

      if (argc == 3)  //IP is provided at command line
      {
         if (strlen(argv[2])>=20)
         {
           cerr << "Too long address!!";
           exit(1);
         }
         strcpy(serv_host_addr, argv[2]);
      }
  }
```

```
// ---------------------------------------------------------
// Initialization:
// ---------------------------------------------------------
//cout << "Client: creating socket\n";
if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
{
   cerr << "Client: socket failed: "<< errno <<"\n";
   exit(1);
}

// cout << "client: binding my local socket\n";  // this is optional
memset( &my_addr, 0, sizeof(my_addr));       // Zero out structure
my_addr.sin_family = AF_INET;                // Internet address family
my_addr.sin_port = htons(0);                 // My port, 0 means any random available one
my_addr.sin_addr.s_addr = htonl(INADDR_ANY); // Any incoming interface

// ---------------------------------------------------------
// binding:
// ---------------------------------------------------------
if ( (bind(sockfd, (struct sockaddr *) &my_addr, sizeof(my_addr)) < 0) )
{
   cerr << "client: bind fail: " << errno << "\n";
   exit(2);
}


// ---------------------------------------------------------
// prepare server address
// ---------------------------------------------------------
// cout << "Client: creating addr structure for server\n";
memset( &server_addr, 0, sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(server_port_id);
//server_addr.sin_addr.s_addr = inet_addr(serv_host_addr);// old style
if (inet_aton(serv_host_addr, &(server_addr.sin_addr))==0) // get server addr
{  // invalid server address
   cerr << "Client: Invalid server address...\n";
   exit(2);
}

// ---------------------------------------------------------
// Message Preparation
// ---------------------------------------------------------
// cout << "Client: initializing message and sending\n";

//strcpy(msg,  "Hello world");
cout << "Input message to server: ";
cin.getline(msg, MESSAGESIZE,'\n');
cout << msg << "  size: " << strlen(msg) << "\n";
```

```
  // ------------------------------------------------------
  // Transmission
  // ------------------------------------------------------
  retcode = sendto(sockfd, msg, strlen(msg)+1, 0,
              (struct sockaddr *) &server_addr, sizeof(server_addr));
  if (retcode <= -1)
  {
     cerr << "client: sendto failed: " << errno << "\n";
     exit(3);
  }

  // ------------------------------------------------------
  // Get message from server
  // ------------------------------------------------------
  addrlen = sizeof(server_addr);   // specify the size of server_addr, this is important !!!!!
  nread = recvfrom(sockfd,msg, MESSAGESIZE, 0,
              (struct sockaddr *) &server_addr, (socklen_t *) &addrlen);
  if (nread >0)
  {
     cout << "Client: relay's response message is: ";
     cout.write(msg, nread);
     cout << "\n";
  }


  // ------------------------------------------------------
  // Termination
  // ------------------------------------------------------
  /* close socket */
  close(sockfd);
}
```