# Server theo giao thức TCP

|  |
| --- |
| // VD1.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #include "winsock2.h"  int main()  {  WSADATA wsaData;  WORD version = MAKEWORD(2, 2);  int res = WSAStartup(version, &wsaData);  if (res == 0)  printf("OK");  else {  printf("Failed");  return 0;  }    SOCKET listeningSocket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN serverAddr;  serverAddr.sin\_family = AF\_INET;  serverAddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  serverAddr.sin\_port = htons(8000);    bind(listeningSocket, (SOCKADDR \*)&serverAddr, sizeof(serverAddr));  listen(listeningSocket, 5);  printf("Waiting for client...\n");  SOCKET clientSocket = accept(listeningSocket, NULL, NULL);  printf("Client connected %d\n", clientSocket);  char \* msg = "Hello client!";  res = send(clientSocket, msg, strlen(msg), 0);  if (res != SOCKET\_ERROR)  printf("%d bytes are sent\n", res);  else  printf("Sending failed");  printf("Waiting for data from client");  char buf[256];  while (true) {  res = recv(clientSocket, buf, sizeof(buf), 0);  if (res <= 0)  break;  buf[res] = 0;  printf("%s\n", buf);  }    system("PAUSE");  closesocket(clientSocket);  closesocket(listeningSocket);  WSACleanup();  return 0;  } |

# PHÂN GIẢI TÊN MIỀN SỬ DỤNG LỆNH getaddrinfo()

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <winsock2.h>  #include <ws2tcpip.h>  /\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/  int main(int argc, char \*argv[]) {  WSADATA wsaData;  WORD version = MAKEWORD(2, 2);  int res = WSAStartup(version, &wsaData);  if (res == 0)  printf("OK");  else  printf("Failed");  struct addrinfo \* result;  SOCKADDR\_IN address;    res = getaddrinfo("www.google.com.vn", "http", NULL, &result);  if (res == 0) {  printf("Phan giai ten mien thanh cong\n");  memcpy(&address, result->ai\_addr, result->ai\_addrlen);  printf("IP: %s", inet\_ntoa(address.sin\_addr));  } else  printf("Phan giai ten mien that bai");  system("PAUSE");  return 0;  } |

# BT6

|  |
| --- |
| // BT6.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #include "winsock2.h"  DWORD WINAPI ClientThread(LPVOID);  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET listener = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  addr.sin\_port = htons(9000);  bind(listener, (SOCKADDR \*)&addr, sizeof(addr));  listen(listener, 5);  while (true)  {  SOCKET client = accept(listener, NULL, NULL);  CreateThread(0, 0, ClientThread, &client, 0, 0);  }  closesocket(listener);  WSACleanup();  return 0;  }  DWORD WINAPI ClientThread(LPVOID lpParam)  {  SOCKET client = \*(SOCKET \*)lpParam;  char buf[1024];  int res;  char cmd[16], tmp[16];  float f1, f2;  while (true)  {  res = recv(client, buf, sizeof(buf), 0);  if (res == SOCKET\_ERROR || res == 0)  break;  buf[res] = 0;  puts(buf);  res = sscanf(buf, "%s %f %f %s", cmd, &f1, &f2, tmp);  if (res == 3)  {  if (strcmp(cmd, "ADD") == 0)  {  float f = f1 + f2;  sprintf(buf, "OK %.2f\n", f);  send(client, buf, strlen(buf), 0);  }  else if (strcmp(cmd, "SUB") == 0)  {  float f = f1 - f2;  sprintf(buf, "OK %.2f\n", f);  send(client, buf, strlen(buf), 0);  }  else if (strcmp(cmd, "MUL") == 0)  {  float f = f1 \* f2;  sprintf(buf, "OK %.2f\n", f);  send(client, buf, strlen(buf), 0);  }  else if (strcmp(cmd, "DIV") == 0)  {  if (f2 == 0)  {  char \*msg = "ERROR Khong the chia cho 0. Hay nhap lai\n";  send(client, msg, strlen(msg), 0);  }  else  {  float f = f1 / f2;  sprintf(buf, "OK %.2f\n", f);  send(client, buf, strlen(buf), 0);  }  }  else  {  char \*msg = "ERROR Sai cu phap. Hay nhap lai\n";  send(client, msg, strlen(msg), 0);  }  }  else  {  char \*msg = "ERROR Sai cu phap. Hay nhap lai\n";  send(client, msg, strlen(msg), 0);  }  }  closesocket(client);  return 0;  } |

# simpleClient

|  |
| --- |
| // SimpleClient.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS  #include "winsock2.h"  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET clientSocket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");  addr.sin\_port = htons(8888);  int res = connect(clientSocket, (SOCKADDR \*)&addr, sizeof(addr));  if (res == SOCKET\_ERROR) {  printf("Connect failed.");  system("PAUSE");  return 0;  }  char \* msg = "Hello Server.\n";  send(clientSocket, msg, strlen(msg), 0);    char buf[256];  while (true) {  printf("Enter a message:");  gets\_s(buf);  // chen ky tu xuong dong  int len = strlen(buf);  buf[len] = '\n';  buf[len + 1] = 0;  send(clientSocket, buf, strlen(buf), 0);  if (strncmp(buf, "exit", 4) == 0)  break;  res = recv(clientSocket, buf, sizeof(buf), 0);  if (res == SOCKET\_ERROR)  break;  buf[res] = 0;  printf("%s", buf);  }  system("PAUSE");  closesocket(clientSocket);  WSACleanup();  return 0;  } |

# simpleHTTPClient

|  |
| --- |
| // SimpleHTTPClient.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS  #include "winsock2.h"  #include "ws2tcpip.h"  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET clientSocket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  /\*addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = inet\_addr("118.102.1.119");  addr.sin\_port = htons(80);\*/  addrinfo \* result;  int res = getaddrinfo("news.zing.vn", "http", NULL, &result);  if (res != 0) {  printf("Khong phan giai duoc dia chi");  system("PAUSE");  return 0;  }  memcpy(&addr, result->ai\_addr, result->ai\_addrlen);  res = connect(clientSocket, (SOCKADDR \*)&addr, sizeof(addr));  if (res == SOCKET\_ERROR) {  printf("Connect failed.");  system("PAUSE");  return 0;  }  char \* msg = "GET / HTTP/1.1\nHost: news.zing.vn\n\n";  send(clientSocket, msg, strlen(msg), 0);  char buf[1024];  while (true) {  res = recv(clientSocket, buf, sizeof(buf), 0);  if (res == SOCKET\_ERROR || res == 0)  break;  if (res < sizeof(buf))  buf[res] = 0;  printf("%s", buf);  }    system("PAUSE");  closesocket(clientSocket);  WSACleanup();  return 0;  } |

# TextClient

|  |
| --- |
| // TextClient.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS  #include "winsock2.h"  #include "ws2tcpip.h"  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET clientSocket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");  addr.sin\_port = htons(8888);  system("PAUSE");  int res = connect(clientSocket, (SOCKADDR \*)&addr, sizeof(addr));  if (res == SOCKET\_ERROR) {  printf("Connect failed.");  system("PAUSE");  return 0;  }  char \* msg = "Hello Server.";  send(clientSocket, msg, strlen(msg), 0);  msg = "Xin chao.";  send(clientSocket, msg, strlen(msg), 0);  char buf[256];  while (true) {  printf("Enter a message:");  gets\_s(buf);  // chen ky tu xuong dong  int len = strlen(buf);  buf[len] = '\n';  buf[len + 1] = 0;  send(clientSocket, buf, strlen(buf), 0);  if (strncmp(buf, "exit", 4) == 0)  break;  }  system("PAUSE");  closesocket(clientSocket);  WSACleanup();  return 0;  } |

# TextServer

|  |
| --- |
| // TextServer.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #include "winsock2.h"  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET listening = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  addr.sin\_port = htons(8888);  bind(listening, (SOCKADDR \*)&addr, sizeof(addr));  listen(listening, 5);  printf("Waiting for client...\n");  SOCKET client = accept(listening, NULL, NULL);  int res;  char buf[1024];  while (true)  {  res = recv(client, buf, sizeof(buf), 0);  if (res == SOCKET\_ERROR || res == 0)  break;  printf("%d bytes received\n", res);  buf[res] = 0;  printf("%s", buf);  }  closesocket(client);  closesocket(listening);  WSACleanup();  return 0;  } |

# FileClient

|  |
| --- |
| // FileClient.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS  #include "winsock2.h"  #include "ws2tcpip.h"  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET clientSocket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");  addr.sin\_port = htons(8888);  system("PAUSE");  int res = connect(clientSocket, (SOCKADDR \*)&addr, sizeof(addr));  if (res == SOCKET\_ERROR) {  printf("Connect failed.");  system("PAUSE");  return 0;  }  FILE \* f;  fopen\_s(&f, "D:\\Test\\sender.png", "rb");  char buf[1024];  while (true) {  res = fread(buf, 1, sizeof(buf), f);  if (res == 0)  break;  send(clientSocket, buf, res, 0);  if (res < 1024)  break;  }  fclose(f);  system("PAUSE");  closesocket(clientSocket);  WSACleanup();  return 0;  } |

# FileServer

|  |
| --- |
| // FileServer.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #include "winsock2.h"  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET listening = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  addr.sin\_port = htons(8888);  bind(listening, (SOCKADDR \*)&addr, sizeof(addr));  listen(listening, 5);  printf("Waiting for client...\n");  SOCKET client = accept(listening, NULL, NULL);  FILE \* f;  fopen\_s(&f, "D:\\Test\\receiver.png", "wb");  int res;  char buf[1024];  while (true)  {  res = recv(client, buf, sizeof(buf), 0);  if (res == SOCKET\_ERROR || res == 0)  break;  fwrite(buf, 1, res, f);  }  fclose(f);  closesocket(client);  closesocket(listening);  WSACleanup();  return 0;  } |

# NumberClient

|  |
| --- |
| // NumberClient.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS  #include "winsock2.h"  #include "ws2tcpip.h"  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET clientSocket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");  addr.sin\_port = htons(8888);  system("PAUSE");  int res = connect(clientSocket, (SOCKADDR \*)&addr, sizeof(addr));  if (res == SOCKET\_ERROR) {  printf("Connect failed.");  system("PAUSE");  return 0;  }  char buf[256];  int n;  while (true) {  printf("Enter a number:");  scanf\_s("%d", &n);  /\*buf[0] = n & 0xFF;  buf[1] = (n >> 8) & 0xFF;  buf[2] = (n >> 16) & 0xFF;  buf[3] = (n >> 24) & 0xFF;  printf("%d %d %d %d\n", (unsigned char)buf[3],  (unsigned char)buf[2], (unsigned char)buf[1], (unsigned char)buf[0]);  send(clientSocket, buf, 4, 0);\*/  send(clientSocket, (char \*)&n, sizeof(n), 0);  if (n == 0)  break;  }  system("PAUSE");  closesocket(clientSocket);  WSACleanup();  return 0;  } |

# NumberServer

|  |
| --- |
| // NumberServer.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #include "winsock2.h"  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET listening = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  addr.sin\_port = htons(8888);  bind(listening, (SOCKADDR \*)&addr, sizeof(addr));  listen(listening, 5);  printf("Waiting for client...\n");  SOCKET client = accept(listening, NULL, NULL);  int res;  char buf[1024];  while (true)  {  res = recv(client, buf, sizeof(buf), 0);  if (res == SOCKET\_ERROR || res == 0)  break;  printf("%d bytes received\n", res);  printf("%d %d %d %d\n", (unsigned char)buf[3],  (unsigned char)buf[2], (unsigned char)buf[1], (unsigned char)buf[0]);  /\*int n = (unsigned char)buf[0] | ((unsigned char)buf[1] << 8) |  ((unsigned char)buf[2] << 16) | ((unsigned char)buf[3] << 24);\*/    int n;  memcpy(&n, buf, 4);  printf("n = %d\n", n);  }  closesocket(client);  closesocket(listening);  WSACleanup();  return 0;  } |

# ClientInfo

|  |
| --- |
| // ClientInfo.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS  #include "winsock2.h"  #include "windows.h"  int main()  {  char buf[512];  int offset = 0;  char cname[256];  DWORD size;  GetComputerNameA(cname, &size);  memcpy(buf + offset, &size, 4);  offset += 4;  memcpy(buf + offset, cname, size);  offset += size;  char driveNames[256];  int res = GetLogicalDriveStringsA(sizeof(driveNames), driveNames);  int numDrives = res / 4;  memcpy(buf + offset, &numDrives, 4);  offset += 4;  DWORD bytePerSector, sectorPerCluster, freeClusters, totalClusters;  for (int i = 0; i < numDrives; i++)  {  GetDiskFreeSpaceA(&driveNames[i \* 4], &sectorPerCluster, &bytePerSector, &freeClusters, &totalClusters);  float totalGBs = (float)totalClusters \* (float)sectorPerCluster \* (float)bytePerSector  / 1024.0 / 1024.0 / 1024.0;  memcpy(buf + offset, &driveNames[i \* 4], 4);  offset += 4;  memcpy(buf + offset, &totalGBs, 4);  offset += 4;  }    WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET client = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");  addr.sin\_port = htons(9000);  system("pause");  connect(client, (SOCKADDR \*)&addr, sizeof(addr));  send(client, buf, offset, 0);  closesocket(client);  WSACleanup();  return 0;  } |

# ServerInfo

|  |
| --- |
| // ServerInfo.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #include "winsock2.h"  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET listener = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  addr.sin\_port = htons(9000);  bind(listener, (SOCKADDR \*)&addr, sizeof(addr));  listen(listener, 5);  SOCKET client = accept(listener, NULL, NULL);  char buf[256];  int res = recv(client, buf, sizeof(buf), 0);  int offset = 0;  int size;  memcpy(&size, buf + offset, 4);  offset += 4;  char cname[256];  memcpy(cname, buf + offset, size);  offset += size;  cname[size] = 0;  printf("Computer name: %s\n", cname);  int numDrives;  memcpy(&numDrives, buf + offset, 4);  offset += 4;  for (int i = 0; i < numDrives; i++)  {  char driveName[16];  float totalGBs;  memcpy(driveName, buf + offset, 4);  offset += 4;  memcpy(&totalGBs, buf + offset, 4);  offset += 4;  printf("%s %f", driveName, totalGBs);  }  closesocket(client);  closesocket(listener);  WSACleanup();  system("pause");  return 0;  } |

# SimpleClient

|  |
| --- |
| // SimpleClient.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS  #include "winsock2.h"  DWORD WINAPI ReceiverThread(LPVOID lpParam);  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET client = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");  addr.sin\_port = htons(9000);  connect(client, (SOCKADDR \*)&addr, sizeof(addr));  CreateThread(0, 0, ReceiverThread, &client, 0, 0);  char buf[256];  while (true)  {  printf("Enter message: ");  gets\_s(buf, sizeof(buf));  send(client, buf, strlen(buf), 0);  }  closesocket(client);  WSACleanup();  return 0;  }  DWORD WINAPI ReceiverThread(LPVOID lpParam)  {  SOCKET client = \*((SOCKET \*)lpParam);  char buf[256];  while (true) {  int res = recv(client, buf, sizeof(buf), 0);  if (res == SOCKET\_ERROR || res == 0)  break;  buf[res] = 0;  puts(buf);  }  return 0;  } |

# SimpleServer

|  |
| --- |
| // SimpleServer.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #include "winsock2.h"  DWORD WINAPI ClientThread(LPVOID lpParams);  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET listener = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  addr.sin\_port = htons(9000);  bind(listener, (SOCKADDR \*)&addr, sizeof(addr));  listen(listener, 5);  SOCKET clients[64];  int numClients = 0;  char \* msg = "New client connected\n";  while (true) {  printf("Waiting for client...\n");  SOCKET client = accept(listener, NULL, NULL);  printf("New client connected: %d\n", client);  for (int i = 0; i < numClients; i++)  send(clients[i], msg, strlen(msg), 0);  clients[numClients] = client;  numClients++;    CreateThread(0, 0, ClientThread, &client, 0, 0);  }    closesocket(listener);  WSACleanup();  return 0;  }  DWORD WINAPI ClientThread(LPVOID lpParams)  {  SOCKET client = \*((SOCKET \*)lpParams);  char buf[256];  while (true)  {  int res = recv(client, buf, sizeof(buf), 0);  if (res == SOCKET\_ERROR || res == 0)  break;  buf[res] = 0;  puts(buf);  }  } |

# CMD Test

|  |
| --- |
| // CMDTest.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  int main(int argc, char \* argv[])  {  printf("Number of params: %d\n", argc);  for (int i = 0; i < argc; i++)  printf("%s\n", argv[i]);  return 0;  } |

# UDPReceiver

|  |
| --- |
| // UDPReceiver.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #include "winsock2.h"  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET receiver = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  addr.sin\_port = htons(9000);  bind(receiver, (SOCKADDR \*)&addr, sizeof(addr));  char buf[256];  int res = recvfrom(receiver, buf, sizeof(buf), 0, NULL, NULL);  buf[res] = 0;  puts(buf);  closesocket(receiver);  WSACleanup();  system("pause");  return 0;  } |

# UDPSender

|  |
| --- |
| // UDPSender.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS  #include "winsock2.h"  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET sender = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");  addr.sin\_port = htons(9000);  char \*msg = "This is UDP sender";  system("pause");  sendto(sender, msg, strlen(msg), 0, (SOCKADDR \*)&addr, sizeof(addr));  closesocket(sender);  WSACleanup();  return 0;  } |

# SelectClient

|  |
| --- |
| // SelectClient.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS  #include "winsock2.h"  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET client = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");  addr.sin\_port = htons(8888);  connect(client, (SOCKADDR \*)&addr, sizeof(addr));  fd\_set fread;  int res;  char buf[256];  while (true)  {  FD\_ZERO(&fread);  FD\_SET(client, &fread);  res = select(0, &fread, NULL, NULL, NULL);  if (res == SOCKET\_ERROR)  break;  if (res > 0)  {  if (FD\_ISSET(client, &fread))  {  res = recv(client, buf, sizeof(buf), 0);  buf[res] = 0;  printf("%s", buf);  }  }  }  closesocket(client);  WSACleanup();  return 0;  } |

# SelectSerVer

|  |
| --- |
| // SelectServer.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #include "winsock2.h"  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET listener = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  addr.sin\_port = htons(8888);  bind(listener, (SOCKADDR \*)&addr, sizeof(addr));  listen(listener, 5);  fd\_set fdread;  int res;  SOCKET clients[64];  int numClients = 0;  char buf[256];  while (true)  {  FD\_ZERO(&fdread);    FD\_SET(listener, &fdread);  for (int i = 0; i < numClients; i++)  FD\_SET(clients[i], &fdread);  res = select(0, &fdread, NULL, NULL, NULL);  if (res == SOCKET\_ERROR)  break;  if (res > 0)  {  if (FD\_ISSET(listener, &fdread))  {  SOCKET client = accept(listener, NULL, NULL);  printf("Client accepted: %d\n", client);  clients[numClients] = client;  numClients++;  }  for (int i = 0; i < numClients; i++)  if (FD\_ISSET(clients[i], &fdread))  {  res = recv(clients[i], buf, sizeof(buf), 0);  if (res == SOCKET\_ERROR || res == 0)  continue;  buf[res] = 0;  printf("%s", buf);  }  }  }  closesocket(listener);  WSACleanup();  return 0;  } |

# Server1

|  |
| --- |
| // Server1.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #include "winsock2.h"  DWORD WINAPI ClientThread(LPVOID);  SOCKET clients[64];  int numClients;  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET listener = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  addr.sin\_port = htons(8888);  bind(listener, (SOCKADDR \*)&addr, sizeof(addr));  listen(listener, 5);  while (true)  {  printf("Waiting for client...\n");  SOCKET client = accept(listener, NULL, NULL);  printf("Client accepted: %d\n", client);  CreateThread(0, 0, ClientThread, &client, 0, 0);  }  closesocket(listener);  WSACleanup();  return 0;  }  void RemoveClient(SOCKET client)  {  int i = 0;  for (; i < numClients; i++)  if (clients[i] == client)  break;  if (i < numClients - 1)  clients[i] = clients[numClients - 1];  numClients--;  }  DWORD WINAPI ClientThread(LPVOID lpParam)  {  SOCKET client = \*(SOCKET \*)lpParam;  char buf[256], sendbuf[256];  int res;  char cmd[16], id[64], tmp[64];  while (true)  {  res = recv(client, buf, sizeof(buf), 0);  if (res == SOCKET\_ERROR || res == 0)  {  closesocket(client);  return 1;  }  // Xu ly buffer  buf[res] = 0;  res = sscanf(buf, "%s %s %s", cmd, id, tmp);  if (res != 2)  {  char \* msg = "Wrong format. Please send again.\n";  send(client, msg, strlen(msg), 0);  }  else  {  if (strcmp(cmd, "client\_id:") != 0)  {  char \* msg = "Wrong format. Please send again.\n";  send(client, msg, strlen(msg), 0);  }  else  {  // Correct format  char \* msg = "OK. You can send message now.\n";  send(client, msg, strlen(msg), 0);  clients[numClients] = client;  numClients++;  break;  }  }  }  while (true)  {  res = recv(client, buf, sizeof(buf), 0);  if (res == SOCKET\_ERROR || res == 0)  {  RemoveClient(client);  closesocket(client);  return 1;  }  // Xu ly buffer  buf[res] = 0;  sprintf(sendbuf, "%s: %s", id, buf);  for (int i = 0; i < numClients; i++)  if (client != clients[i])  send(clients[i], sendbuf, strlen(sendbuf), 0);  }  return 0;  } |

# Server2

|  |
| --- |
| // Server2.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #include "winsock2.h"  DWORD WINAPI ClientThread(LPVOID);  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET listener = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  addr.sin\_port = htons(8888);  bind(listener, (SOCKADDR \*)&addr, sizeof(addr));  listen(listener, 5);  while (true)  {  printf("Waiting for client...\n");  SOCKET client = accept(listener, NULL, NULL);  printf("Client accepted: %d\n", client);  CreateThread(0, 0, ClientThread, &client, 0, 0);  }  closesocket(listener);  WSACleanup();  return 0;  }  DWORD WINAPI ClientThread(LPVOID lpParam)  {  SOCKET client = \*(SOCKET \*)lpParam;  char buf[256], filebuf[256], cmdbuf[256];  int res;  char user[64], pass[64], tmp[64];  while (true)  {  res = recv(client, buf, sizeof(buf), 0);  if (res == SOCKET\_ERROR || res == 0)  {  closesocket(client);  return 1;  }  // Xu ly buffer  buf[res] = 0;  res = sscanf(buf, "%s %s %s", user, pass, tmp);  if (res != 2)  {  char \* msg = "Wrong format. Please send again.\n";  send(client, msg, strlen(msg), 0);  }  else  {  sprintf(buf, "%s %s", user, pass);  int found = 0;  FILE \*f = fopen("D:\\Test\\data.txt", "r");  while (fgets(filebuf, sizeof(filebuf), f) != NULL)  {  filebuf[strlen(filebuf) - 1] = 0;  if (strcmp(buf, filebuf) == 0)  {  found = 1;  break;  }  }  fclose(f);  if (found == 0)  {  char \* msg = "Wrong username or password. Please send again.\n";  send(client, msg, strlen(msg), 0);  }  else  break;  }  }  while (true)  {  res = recv(client, buf, sizeof(buf), 0);  if (res == SOCKET\_ERROR || res == 0)  {  closesocket(client);  return 1;  }  // Xu ly buffer  buf[res - 1] = 0;  sprintf(cmdbuf, "%s > D:\\Test\\out.txt", buf);  system(cmdbuf);  FILE \*f = fopen("D:\\Test\\out.txt", "r");  while (fgets(filebuf, sizeof(filebuf), f) != NULL)  send(client, filebuf, strlen(filebuf), 0);  fclose(f);  }  return 0;  } |

# BT8

|  |
| --- |
| // BT8.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #include "winsock.h"  DWORD WINAPI ClientThread(LPVOID);  void RemoveClient(SOCKET);  SOCKET clients[64];  char \*ids[64];  int numClients = 0;  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  addr.sin\_port = htons(9000);  SOCKET listener = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  bind(listener, (SOCKADDR \*)&addr, sizeof(addr));  listen(listener, 5);  while (1)  {  SOCKET client = accept(listener, NULL, NULL);  CreateThread(0, 0, ClientThread, &client, 0, 0);  printf("New client accepted: %d", client);  }  return 0;  }  DWORD WINAPI ClientThread(LPVOID lpParam)  {  SOCKET client = \*(SOCKET \*)lpParam;  char buf[1024], sendbuf[1024];  int res;  int i, j;  char cmd[16], id[64], tmp[64];  while (1)  {  res = recv(client, buf, sizeof(buf), 0);  if (res == SOCKET\_ERROR || res == 0)  {  RemoveClient(client);  closesocket(client);  return 1;  }  buf[res] = 0;  for (i = 0; i < numClients; i++)  if (clients[i] == client) break;  if (i == numClients)  {  // Chua dang nhap  res = sscanf(buf, "%s %s %s", cmd, id, tmp);  if (res == 2 && strcmp(cmd, "connect") == 0)  {  clients[numClients] = client;  ids[numClients] = (char \*)malloc(strlen(id) + 1);  memcpy(ids[numClients], id, strlen(id) + 1);  numClients++;  }  else  {  char \* msg = "ERROR Sai cu phap\n";  send(client, msg, strlen(msg), 0);  }  }  else  {  // Da dang nhap  res = sscanf(buf, "%s %s %s", cmd, id, tmp);  if (res == 1 && strcmp(cmd, "list") == 0)  {  for (j = 0; j < numClients; j++)  {  send(client, ids[j], strlen(ids[j]), 0);  send(client, " ", 1, 0);  }  send(client, "\n", 1, 0);  }  else if (res == 3 && strcmp(cmd, "SEND"))  {  if (strcmp(id, "ALL") == 0)  {  sprintf(sendbuf, "%s %s\n", ids[i], buf + strlen(cmd) + strlen(id) + 2);  for (j = 0; j < numClients; j++)  if (clients[j] != client)  send(clients[j], sendbuf, strlen(sendbuf), 0);  }  else  {  for (j = 0; j < numClients; j++)  if (strcmp(ids[j], id) == 0) break;  if (j < numClients)  {  sprintf(sendbuf, "%s %s\n", ids[i], buf + strlen(cmd) + strlen(id) + 2);  send(clients[j], sendbuf, strlen(sendbuf), 0);  }  else  {  char \* msg = "ERROR Khong tim duoc id\n";  send(client, msg, strlen(msg), 0);  }  }  }  else if (res == 1 && strcmp(cmd, "disconnect") == 0)  {  RemoveClient(client);  }  else  {  char \* msg = "ERROR Sai cu phap\n";  send(client, msg, strlen(msg), 0);  }  }    }  }  void RemoveClient(SOCKET client)  {  int i = 0;  for (; i < numClients; i++)  if (clients[i] == client) break;  if (i < numClients)  {  if (i < numClients - 1)  {  clients[i] = clients[numClients - 1];  ids[i] = ids[numClients - 1];  }  numClients--;  }  } |

# Client1

|  |
| --- |
| // Client1.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS  #include "winsock2.h"  #define WM\_SOCKET WM\_USER + 1  BOOL CALLBACK WinProc(HWND, UINT, WPARAM, LPARAM);  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET client = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");  addr.sin\_port = htons(8000);  connect(client, (SOCKADDR \*)&addr, sizeof(addr));  WNDCLASS wndclass;  CHAR \*providerClass = "AsyncSelect";  HWND window;  wndclass.style = 0;  wndclass.lpfnWndProc = (WNDPROC)WinProc;  wndclass.cbClsExtra = 0;  wndclass.cbWndExtra = 0;  wndclass.hInstance = NULL;  wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);  wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);  wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);  wndclass.lpszMenuName = NULL;  wndclass.lpszClassName = (LPCWSTR)providerClass;  if (RegisterClass(&wndclass) == 0)  return NULL;  // Create a window  if ((window = CreateWindow((LPCWSTR)providerClass, L"", WS\_OVERLAPPEDWINDOW,  CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT,  NULL, NULL, NULL, NULL)) == NULL)  return NULL;  WSAAsyncSelect(client, window, WM\_SOCKET, FD\_READ | FD\_CLOSE);  MSG msg;  while (GetMessage(&msg, NULL, 0, 0) > 0)  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  return 0;  }  BOOL CALLBACK WinProc(HWND hDlg, UINT wMsg, WPARAM wParam, LPARAM lParam)  {  if (wMsg == WM\_SOCKET)  {  if (WSAGETSELECTERROR(lParam))  {  closesocket((SOCKET)wParam);  return TRUE;  }  if (WSAGETSELECTEVENT(lParam) == FD\_READ)  {  char buf[256];  int res = recv((SOCKET)wParam, buf, sizeof(buf), 0);  buf[res] = 0;  printf("%s", buf);  }  else if (WSAGETSELECTEVENT(lParam) == FD\_CLOSE) {  closesocket((SOCKET)wParam);  return TRUE;  }  }  } |

# Server1

|  |
| --- |
| // Server1.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS  #include "winsock2.h"  #define WM\_SOCKET WM\_USER + 1  SOCKET registeredClients[64];  int numRegisteredClients = 0;  char cmd[16], id[64], tmp[64];  char \* ids[64];  char sendbuf[1024];  BOOL CALLBACK WinProc(HWND, UINT, WPARAM, LPARAM);  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET listener = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  addr.sin\_port = htons(8000);  bind(listener, (SOCKADDR \*)&addr, sizeof(addr));  listen(listener, 5);  WNDCLASS wndclass;  CHAR \*providerClass = "AsyncSelect";  HWND window;  wndclass.style = 0;  wndclass.lpfnWndProc = (WNDPROC)WinProc;  wndclass.cbClsExtra = 0;  wndclass.cbWndExtra = 0;  wndclass.hInstance = NULL;  wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);  wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);  wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);  wndclass.lpszMenuName = NULL;  wndclass.lpszClassName = (LPCWSTR)providerClass;  if (RegisterClass(&wndclass) == 0)  return NULL;  // Create a window  if ((window = CreateWindow((LPCWSTR)providerClass, L"", WS\_OVERLAPPEDWINDOW,  CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT,  NULL, NULL, NULL, NULL)) == NULL)  return NULL;  WSAAsyncSelect(listener, window, WM\_SOCKET, FD\_ACCEPT);  MSG msg;  while (GetMessage(&msg, NULL, 0, 0) > 0)  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  return 0;  }  BOOL CALLBACK WinProc(HWND hDlg, UINT wMsg, WPARAM wParam, LPARAM lParam)  {  if (wMsg == WM\_SOCKET)  {  if (WSAGETSELECTERROR(lParam))  {  closesocket((SOCKET)wParam);  return TRUE;  }  if (WSAGETSELECTEVENT(lParam) == FD\_ACCEPT)  {  SOCKET client = accept((SOCKET)wParam, NULL, NULL);  char \*msg = "Hello client.\n";  send(client, msg, strlen(msg), 0);  WSAAsyncSelect(client, hDlg, WM\_SOCKET, FD\_READ | FD\_CLOSE);  }  else if (WSAGETSELECTEVENT(lParam) == FD\_READ)  {  char buf[256];  int res = recv((SOCKET)wParam, buf, sizeof(buf), 0);  buf[res] = 0;  printf("%s", buf);  int j = 0;  for (; j < numRegisteredClients; j++)  if (registeredClients[j] == (SOCKET)wParam)  break;  if (j >= numRegisteredClients)  {  // chua dang nhap  res = sscanf(buf, "%s %s %s", cmd, id, tmp);  if (res != 2)  {  char \* msg = "Wrong format. Please send again.\n";  send((SOCKET)wParam, msg, strlen(msg), 0);  }  else  {  if (strcmp(cmd, "client\_id:") != 0)  {  char \* msg = "Wrong format. Please send again.\n";  send((SOCKET)wParam, msg, strlen(msg), 0);  }  else  {  // Correct format  char \* msg = "OK. You can send message now.\n";  send((SOCKET)wParam, msg, strlen(msg), 0);  registeredClients[numRegisteredClients] = (SOCKET)wParam;  ids[numRegisteredClients] = (char \*)malloc(64);  memcpy(ids[numRegisteredClients], id, strlen(id) + 1);  numRegisteredClients++;  }  }  }  else  {  // da dang nhap  sprintf(sendbuf, "%s: %s", ids[j], buf);  for (int j = 0; j < numRegisteredClients; j++)  if (registeredClients[j] != (SOCKET)wParam)  send(registeredClients[j], sendbuf, strlen(sendbuf), 0);  }  }  else if (WSAGETSELECTEVENT(lParam) == FD\_CLOSE)  {  closesocket((SOCKET)wParam);  return TRUE;  }  }  } |

# GUIServer

|  |
| --- |
| // GUIServer.cpp : Defines the entry point for the application.  //  #include "stdafx.h"  #include "GUIServer.h"  #define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS  #include "winsock2.h"  #define MAX\_LOADSTRING 100  #define WM\_SOCKET WM\_USER + 1  SOCKET listener;  // Global Variables:  HINSTANCE hInst; // current instance  WCHAR szTitle[MAX\_LOADSTRING]; // The title bar text  WCHAR szWindowClass[MAX\_LOADSTRING]; // the main window class name  // Forward declarations of functions included in this code module:  ATOM MyRegisterClass(HINSTANCE hInstance);  BOOL InitInstance(HINSTANCE, int);  LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);  INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);  int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,  \_In\_opt\_ HINSTANCE hPrevInstance,  \_In\_ LPWSTR lpCmdLine,  \_In\_ int nCmdShow)  {  UNREFERENCED\_PARAMETER(hPrevInstance);  UNREFERENCED\_PARAMETER(lpCmdLine);  // TODO: Place code here.  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  // Initialize global strings  LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);  LoadStringW(hInstance, IDC\_GUISERVER, szWindowClass, MAX\_LOADSTRING);  MyRegisterClass(hInstance);  // Perform application initialization:  if (!InitInstance (hInstance, nCmdShow))  {  return FALSE;  }  HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_GUISERVER));  MSG msg;  // Main message loop:  while (GetMessage(&msg, nullptr, 0, 0))  {  if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  }  return (int) msg.wParam;  }  //  // FUNCTION: MyRegisterClass()  //  // PURPOSE: Registers the window class.  //  ATOM MyRegisterClass(HINSTANCE hInstance)  {  WNDCLASSEXW wcex;  wcex.cbSize = sizeof(WNDCLASSEX);  wcex.style = CS\_HREDRAW | CS\_VREDRAW;  wcex.lpfnWndProc = WndProc;  wcex.cbClsExtra = 0;  wcex.cbWndExtra = 0;  wcex.hInstance = hInstance;  wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_GUISERVER));  wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);  wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);  wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_GUISERVER);  wcex.lpszClassName = szWindowClass;  wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));  return RegisterClassExW(&wcex);  }  //  // FUNCTION: InitInstance(HINSTANCE, int)  //  // PURPOSE: Saves instance handle and creates main window  //  // COMMENTS:  //  // In this function, we save the instance handle in a global variable and  // create and display the main program window.  //  BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)  {  hInst = hInstance; // Store instance handle in our global variable  HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,  CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);  if (!hWnd)  {  return FALSE;  }  ShowWindow(hWnd, nCmdShow);  UpdateWindow(hWnd);  return TRUE;  }  int startServer(HWND hWnd)  {  listener = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  addr.sin\_port = htons(9000);  bind(listener, (SOCKADDR \*)&addr, sizeof(addr));  listen(listener, 5);  WSAAsyncSelect(listener, hWnd, WM\_SOCKET, FD\_ACCEPT);  return 0;  }  int stopServer()  {  closesocket(listener);  return 0;  }  //  // FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)  //  // PURPOSE: Processes messages for the main window.  //  // WM\_COMMAND - process the application menu  // WM\_PAINT - Paint the main window  // WM\_DESTROY - post a quit message and return  //  //  LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)  {  switch (message)  {  case WM\_SOCKET:  {  if (WSAGETSELECTERROR(lParam))  {  closesocket((SOCKET)wParam);  break;  }  if (WSAGETSELECTEVENT(lParam) == FD\_ACCEPT)  {  SOCKET client = accept((SOCKET)wParam, NULL, NULL);  char \*msg = "Hello client.\n";  send(client, msg, strlen(msg), 0);  }  }  break;  case WM\_COMMAND:  {  int wmId = LOWORD(wParam);  // Parse the menu selections:  switch (wmId)  {  case IDM\_ABOUT:  DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);  break;  case IDM\_EXIT:  DestroyWindow(hWnd);  break;  case IDM\_START\_SERVER:  startServer(hWnd);  break;  case IDM\_STOP\_SERVER:  stopServer();  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  }  break;  case WM\_PAINT:  {  PAINTSTRUCT ps;  HDC hdc = BeginPaint(hWnd, &ps);  // TODO: Add any drawing code that uses hdc here...  EndPaint(hWnd, &ps);  }  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  return 0;  }  // Message handler for about box.  INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)  {  UNREFERENCED\_PARAMETER(lParam);  switch (message)  {  case WM\_INITDIALOG:  return (INT\_PTR)TRUE;  case WM\_COMMAND:  if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)  {  EndDialog(hDlg, LOWORD(wParam));  return (INT\_PTR)TRUE;  }  break;  }  return (INT\_PTR)FALSE;  } |

# EventSelect

|  |
| --- |
| // VD10\_EventSelect.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #include "winsock2.h"  int main()  {  SOCKET sockets[WSA\_MAXIMUM\_WAIT\_EVENTS];  WSAEVENT events[WSA\_MAXIMUM\_WAIT\_EVENTS];  int totalEvents = 0;  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET listener = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  addr.sin\_port = htons(9000);  bind(listener, (SOCKADDR \*)&addr, sizeof(addr));  listen(listener, 5);  WSAEVENT newEvent = WSACreateEvent();  WSAEventSelect(listener, newEvent, FD\_ACCEPT);  sockets[totalEvents] = listener;  events[totalEvents] = newEvent;  totalEvents++;  WSANETWORKEVENTS networkEvents;  while (true)  {  int index = WSAWaitForMultipleEvents(totalEvents, events, FALSE, 5000, FALSE);  if (index == WSA\_WAIT\_FAILED)  break;  if (index == WSA\_WAIT\_TIMEOUT)  {  printf("Timed out.\n");  continue;  }  index = index - WSA\_WAIT\_EVENT\_0;  for (int i = index; i < totalEvents; i++)  {  index = WSAWaitForMultipleEvents(1, &events[i], FALSE, WSA\_INFINITE, FALSE);  if (index == WSA\_WAIT\_FAILED)  continue;  if (index == WSA\_WAIT\_TIMEOUT)  {  printf("Timed out.\n");  continue;  }  WSAEnumNetworkEvents(sockets[i], events[i], &networkEvents);  if (networkEvents.lNetworkEvents & FD\_ACCEPT)  {  if (networkEvents.iErrorCode[FD\_ACCEPT\_BIT] != 0)  continue;  SOCKET client = accept(sockets[i], NULL, NULL);    if (totalEvents > WSA\_MAXIMUM\_WAIT\_EVENTS)  {  printf("Too many events.\n");  closesocket(client);  continue;  }  newEvent = WSACreateEvent();  WSAEventSelect(client, newEvent, FD\_READ);  sockets[totalEvents] = client;  events[totalEvents] = newEvent;  totalEvents++;  printf("New client accepted: %d", client);  }  if (networkEvents.lNetworkEvents & FD\_READ)  {  if (networkEvents.iErrorCode[FD\_READ\_BIT] != 0)  continue;  char buf[1024];  int res = recv(sockets[i], buf, sizeof(buf), 0);  if (res == SOCKET\_ERROR || res == 0)  continue;    buf[res] = 0;  printf("%s", buf);  }  }  }  closesocket(listener);  WSACleanup();  return 0;  } |

# CompletionRoutine

|  |
| --- |
| // VD\_CompletionRoutine.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #include "winsock2.h"  SOCKET client;  OVERLAPPED overlapped;  char buf[1024];  WSABUF databuf;  DWORD flags = 0;  DWORD bytesReceived = 0;  int ret;  void CALLBACK CompletionRoutine(DWORD, DWORD, LPOVERLAPPED, DWORD);  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET listener = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  addr.sin\_port = htons(9000);  bind(listener, (SOCKADDR \*)&addr, sizeof(addr));  listen(listener, 5);  client = accept(listener, NULL, NULL);  memset(&overlapped, 0, sizeof(overlapped));  databuf.buf = buf;  databuf.len = sizeof(buf);  ret = WSARecv(client, &databuf, 1, &bytesReceived, &flags, &overlapped, CompletionRoutine);  while (1)  SleepEx(WSA\_INFINITE, true);  return 0;  }  void CALLBACK CompletionRoutine(DWORD dwError, DWORD dwBytesReceived, LPOVERLAPPED lpOverlapped, DWORD dwFlags)  {  if (dwError != 0 || dwBytesReceived == 0)  {  closesocket(client);  return;  }  // Xu ly du lieu nhan duoc trong buffer  buf[dwBytesReceived] = 0;  printf("%s", buf);  // Yeu cau nhan du lieu tiep theo  memset(&overlapped, 0, sizeof(overlapped));  flags = 0;  ret = WSARecv(client, &databuf, 1, &bytesReceived, &flags, &overlapped, CompletionRoutine);  if (ret == SOCKET\_ERROR)  {  ret = WSAGetLastError();  if (ret != WSA\_IO\_PENDING)  {  printf("Error: %d", ret);  }  }  return;  } |

# Event

|  |
| --- |
| // VD\_Event.cpp : Defines the entry point for the console application.  //  #include "stdafx.h"  #include "winsock2.h"  int main()  {  WSADATA wsa;  WSAStartup(MAKEWORD(2, 2), &wsa);  SOCKET listener = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  SOCKADDR\_IN addr;  addr.sin\_family = AF\_INET;  addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);  addr.sin\_port = htons(9000);  bind(listener, (SOCKADDR \*)&addr, sizeof(addr));  listen(listener, 5);  SOCKET client = accept(listener, NULL, NULL);  OVERLAPPED overlapped;  memset(&overlapped, 0, sizeof(overlapped));  WSAEVENT receiveEvent = WSACreateEvent();  overlapped.hEvent = receiveEvent;  char buf[1024];  WSABUF databuf;  databuf.buf = buf;  databuf.len = sizeof(buf);  DWORD bytesReceived = 0;  DWORD flags = 0;  int ret;  while (1)  {  ret = WSARecv(client, &databuf, 1, &bytesReceived, &flags, &overlapped, 0);  if (ret == SOCKET\_ERROR)  {  ret = WSAGetLastError();  if (ret == WSA\_IO\_PENDING)  printf("Receiving data...\n");  else  {  printf("Error: %d\n", ret);  continue;  }  }  ret = WSAWaitForMultipleEvents(1, &receiveEvent, FALSE, WSA\_INFINITE, FALSE);  if (ret == WSA\_WAIT\_FAILED)  continue;  WSAResetEvent(receiveEvent);  ret = WSAGetOverlappedResult(client, &overlapped, &bytesReceived, FALSE, &flags);  if (bytesReceived == 0)  break;  buf[bytesReceived] = 0;  printf("%s", buf);  }  closesocket(client);  closesocket(listener);  WSACleanup();  return 0;  } |