20220416-socket

1.过程描述

ServerDemo.cpp C++ 复制代码

```cpp
#include <stdio.h>
#include <WinSock2.h>
#include <list>
#include <algorithm>
#include <string.h>
#pragma comment (lib,"ws2_32.lib")

#define MAXCONN 5
#define BUFLEN 255

using namespace std;

typedef list<SOCKET> ListCONN;
typedef list<SOCKET> ListConErr;

int main(int argc, char* argv[])
{
    WSADATA wsaData;
    int nRC;
    SOCKET servSock;
    SOCKADDR_IN servAddr, clntAddr;
    int nAddrLen = sizeof(SOCKADDR);
    char sendBuf[BUFLEN], recvBuf[BUFLEN];
    ListCONN conList;
    ListCONN::iterator itor;
    ListConErr conErrList;
    ListConErr::iterator itorErr;

    FD_SET rfds, wfds;
    u_long uNonBlock;

    nRC = WSAStartup(MAKEWORD(2, 2), &wsaData);
    if (nRC)
    {
        printf("Server initialize winsock error!\n");
        return 0;
    }
    if (wsaData.wVersion != MAKEWORD(2, 2))
    {
        printf("Server's winsock version error!\n");
        WSACleanup();
        return 0;
    }
    printf("Server's winsock initialized!\n");

```

```
46      servSock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
47      if (servSock == INVALID_SOCKET)
48      {
49          printf("Server create socket error\n");
50          WSACleanup();
51          return 0;
52      }
53      printf("Server TCP socket create OK!\n");
54
55      servAddr.sin_family = AF_INET;
56      servAddr.sin_port = htons(5050);
57      servAddr.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");
58      nRC = bind(servSock, (LPSOCKADDR)&servAddr, sizeof(servAddr));
59      if (nRC == SOCKET_ERROR)
60      {
61          printf("Server socket bind error!\n");
62          closesocket(servSock);
63          WSACleanup();
64          return 0;
65      }
66      printf("Server socket bind OK!\n");
67
68      nRC = listen(servSock, MAXCONN);
69      if (nRC == SOCKET_ERROR)
70      {
71          printf("Server socket listen error!\n");
72          closesocket(servSock);
73          WSACleanup();
74          return 0;
75      }
76
77      uNonBlock = 1;
78      ioctlsocket(servSock, FIONBIO, &uNonBlock);
79      while (1)
80      {
81          for (itorErr = conErrList.begin(); itorErr != conErrList.end();
    itorErr++)
82          {
83              itor = find(conList.begin(), conList.end(), *itorErr);
84              if (itor != conList.end())
85                  conList.erase(itor);
86          }
87
88          FD_ZERO(&rfds);
89          FD_ZERO(&wfds);
90
91          FD_SET(servSock, &rfds);
92
```

```
93          for (itor = conList.begin(); itor != conList.end(); itor++)
94          {
95              uNonBlock = 1;
96              ioctlsocket(*itor, FIONBIO, &uNonBlock);
97              FD_SET(*itor, &rfds);
98              FD_SET(*itor, &wfds);
99          }
100
101         int nTotal = select(0, &rfds, &wfds, NULL, NULL);
102         if (FD_ISSET(servSock, &rfds))
103         {
104             nTotal--;
105             SOCKET connSock = accept(servSock, (LPSOCKADDR)&clntAddr,
    &nAddrLen);
106             if (connSock == INVALID_SOCKET)
107             {
108                 printf("Server accept connection request error!\n");
109                 closesocket(servSock);
110                 WSACleanup();
111                 return 0;
112             }
113             sprintf_s(sendBuf, "来自%s的游客进入聊天室!\n",
    inet_ntoa(clntAddr.sin_addr));
114             printf("%s", sendBuf);
115             conList.insert(conList.end(), connSock);
116         }
117         if (nTotal > 0)
118         {
119             for (itor = conList.begin(); itor != conList.end(); itor++)
120             {
121                 if (FD_ISSET(*itor, &wfds))
122                 {
123                     if (strlen(sendBuf) > 0)
124                     {
125                         nRC = send(*itor, sendBuf, strlen(sendBuf), 0);
126                         if (nRC == SOCKET_ERROR)
127                         {
128                             conErrList.insert(conErrList.end(), *itor);
129                         }
130                         else
131                         {
132                             memset(sendBuf, '\0', BUFLEN);
133                         }
134                     }
135                 }
136                 if (FD_ISSET(*itor, &rfds))
137                 {
138                     nRC = recv(*itor, recvBuf, BUFLEN, 0);
```

```
139                     if (nRC == SOCKET_ERROR)
140                     {
141                         conErrList.insert(conErrList.end(), *itor);
142                     }
143                     else
144                     {
145                         recvBuf[nRC] = '\n';
146                         sprintf_s(sendBuf, "\n游客说:%s\n", recvBuf);
147                         printf("%s", sendBuf);
148                     }
149                 }
150             }
151         }
152     }
153     closesocket(servSock);
154     WSACleanup();
155 
156 }
```

ClientDemo.cpp C++ 复制代码

```cpp
#include <stdio.h>
#include <WinSock2.h>
#include <string.h>
#include <Windows.h>
#include <process.h>
#include <WinBase.h>
#pragma comment (lib,"ws2_32.lib")

#define BUFLEN 255
CRITICAL_SECTION gCriticalSection;

unsigned __stdcall GetInputs(void* arg);

int main(int argc, char* argv[])
{
    WSADATA wsaData;
    int nRC;
    SOCKADDR_IN servAddr, clntAddr;
    SOCKET clientSock;
    char sendBuf[BUFLEN], recvBuf[BUFLEN];
    FD_SET rfds, wfds;
    u_long uNonBlock;
    HANDLE hThread;
    unsigned dwThreadID;

    if (argc!= 2)
    {
        printf("Usage:%s ClientIPAddress name\n", argv[0]);
        return 0;
    }
    InitializeCriticalSection(&gCriticalSection);
    nRC = WSAStartup(MAKEWORD(2, 2), &wsaData);
    if (nRC)
    {
        printf("Client initialize winsock error!\n");
        return 0;
    }
    if (wsaData.wVersion != MAKEWORD(2, 2))
    {
        printf("Client's winsock version error!\n");
        WSACleanup();
        return 0;
    }
    printf("Client's winsock initialized!\n");

```

```
46      clientSock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
47      if (clientSock == INVALID_SOCKET)
48      {
49          printf("Client's create socket error!\n");
50          WSACleanup();
51          return 0;
52      }
53      printf("Client socket create OK!\n");
54
55      clntAddr.sin_family = AF_INET;
56      clntAddr.sin_port = htons(0);
57      clntAddr.sin_addr.S_un.S_addr = inet_addr(argv[1]);
58      nRC = bind(clientSock, (LPSOCKADDR)&clntAddr, sizeof(clntAddr));
59      if (nRC == SOCKET_ERROR)
60      {
61          printf("Client socket bind error!\n");
62          closesocket(clientSock);
63          WSACleanup();
64          return 0;
65      }
66      printf("Client socket bind OK!\n");
67
68      servAddr.sin_family = AF_INET;
69      servAddr.sin_port = htons(5050);
70      servAddr.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");
71
72      nRC = connect(clientSock, (LPSOCKADDR)&servAddr, sizeof(servAddr));
73      if (nRC == SOCKET_ERROR)
74      {
75          printf("连接服务器失败!\n");
76          closesocket(clientSock);
77          WSACleanup();
78          return 0;
79      }
80      hThread = (HANDLE)_beginthreadex(NULL, 0, GetInputs, sendBuf, 0,
    &dwThreadID);
81      while (1)
82      {
83          memset(sendBuf, '\0', BUFLEN);
84          memset(recvBuf, '\0', BUFLEN);
85          FD_ZERO(&rfds);
86          FD_ZERO(&wfds);
87          FD_SET(clientSock, &rfds);
88          FD_SET(clientSock, &wfds);
89          uNonBlock = 1;
90          ioctlsocket(clientSock, FIONBIO, &uNonBlock);
91          select(0, &rfds, &wfds, NULL, NULL);
92
```

```
93          if (FD_ISSET(clientSock, &rfds))
94          {
95              nRC = recv(clientSock, recvBuf, BUFLEN, 0);
96              if (nRC == SOCKET_ERROR)
97              {
98                  printf("接收数据失败!\n");
99                  DeleteCriticalSection(&gCriticalSection);
100                 closesocket(clientSock);
101                 WSACleanup();
102                 return 0;
103             }
104             else if (nRC > 0)
105             {
106                 recvBuf[nRC] = '\0';
107                 printf("\n%s\n", recvBuf);
108             }
109         }
110         if (FD_ISSET(clientSock, &wfds))
111         {
112             if (strlen(sendBuf) > 0)
113             {
114                 nRC = send(clientSock, sendBuf, strlen(sendBuf), 0);
115                 if (nRC == SOCKET_ERROR)
116                 {
117                     printf("发送数据失败!\n");
118                     DeleteCriticalSection(&gCriticalSection);
119                     closesocket(clientSock);
120                     WSACleanup();
121                     return 0;
122                 }
123                 else
124                 {
125                     EnterCriticalSection(&gCriticalSection);
126                     sendBuf[0] = '\0';
127                     LeaveCriticalSection(&gCriticalSection);
128                 }
129             }
130         }
131         if (strcmp(sendBuf, "exit") == 0)
132             break;
133     }
134     DeleteCriticalSection(&gCriticalSection);
135     closesocket(clientSock);
136     WSACleanup();
137 }
138
139 unsigned __stdcall GetInputs(void* arg)
140 {
```

```
141         char* inputs = (char*)arg;
142         while (1)
143         {
144             printf("\n我要发言:");
145             EnterCriticalSection(&gCriticalSection);
146             gets_s(inputs,sizeof(inputs));
147             LeaveCriticalSection(&gCriticalSection);
148             if (strcmp(inputs, "exit") == 0)
149                 return EXIT_SUCCESS;
150         }
151     }
```

2.结果输出

今天只看了华中科技大学的一个SOCKET编程实验，copy了一下里面的示例代码。感觉相较于之前的程序感觉复杂了一些，涉及到非阻塞编程的一些知识，还没完全看懂。明天继续。