# 实验1 socket编程

## 实验目的：

1. 熟悉socket编程的一般流程。
2. 掌握关键的socket API 函数或结构体

## 实验内容：

1. 根据所给程序，编写以一个能够同时响应多个客户端连接请求的服务器
2. 测试、验证是否有效；
3. 客户端向服务器发送的数据格式

{设备号：无符号整型：1字节；

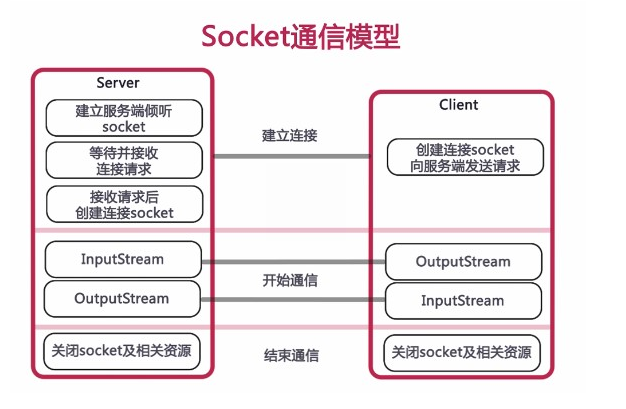
线路号：字符串： 10字节；

电压： 整型： 2字节；

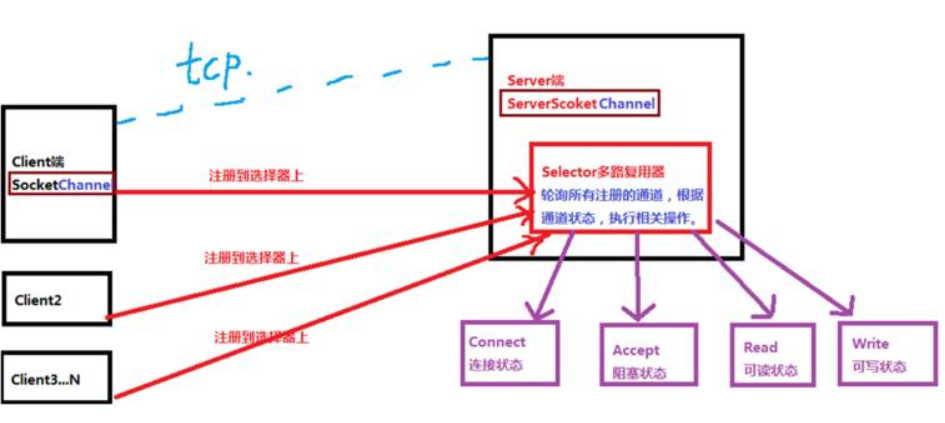
电流： 整型： 2字节}

## 实验步骤：

1. 熟悉socket 服务器端程序的流程：画出客户端与服务器通信的流程



1. 画出异步socket select方法客户端与服务器端通信的流程



3、把所给服务器端程序改写为能够同时响应多个客户端连接请求的服务器程序，写出你实现的过程

// serverDemo.cpp : 此文件包含 "main" 函数。程序执行将在此处开始并结束。

//

#include "pch.h"

#include <iostream>

// server.cpp : Defines the entry point for the console application.

//

#pragma comment(lib,"WS2\_32.lib")

/\* File Name: server.c \*/

#include <winsock2.h>

#include <WS2tcpip.h>

#include <stdlib.h>

#include <stdio.h>

#include <ctime>

#include <time.h>

#include <string.h>

#include <iostream>

#include <unordered\_map>

#include "serverDemo.h"

#define TRUE 1

int main()

{

int sock, length;

struct sockaddr\_in server;

struct sockaddr tcpaddr;

int msgsock;

char buf[1024];

//char buf\_send[1024];

char buf\_time[100];

int rval, len;

unsigned int threadID=0;

\_stSock st\_sock[10];

std::unordered\_map<std::string, pSession> sessionMap;

std::unordered\_map<std::string, bool > client;

WSADATA wsa;

/\* 建立套接字 \*/

if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)/\*#######补充完整#######\*/

{

printf("fail to load socket.");

return 1;

}

sock = socket(AF\_INET, SOCK\_STREAM, 0);

if (sock < 0) {

perror("opening stream socket");

exit(1);

}

/\* 使用任意端口命名套接字 \*/

server.sin\_family = AF\_INET;

server.sin\_addr.s\_addr = htonl(INADDR\_ANY);

server.sin\_port = htons(1234);

if (bind(sock, (struct sockaddr \*)&server, sizeof(server)) < 0) {

perror("binding stream socket");

exit(1);

}

/\* 找出指定的端口号并打印出来 \*/

length = sizeof(server);

if (getsockname(sock, (struct sockaddr \*)&server, &length) < 0) {

perror("0getting socket name");

exit(1);

}

struct addrinfo hints, \*res;

memset(&hints, 0, sizeof(addrinfo));

hints.ai\_family = AF\_INET;

hints.ai\_socktype = SOCK\_STREAM;

int ret = getaddrinfo("DESKTOP-11GNR7U", NULL, &hints, &res);

if (ret < 0)

{

perror("fail to get address");

exit(1);

}

struct sockaddr\_in \*pstAddr = (struct sockaddr\_in \*)(res->ai\_addr);

printf("server IP:%s,socket port #%d\n", inet\_ntoa(pstAddr->sin\_addr), ntohs(server.sin\_port));

/\* 开始接收连接 \*/

listen(sock, 5);

len = sizeof(struct sockaddr);

do {

/\*等待client的连接\*/

msgsock = accept(sock, (struct sockaddr \*)&tcpaddr, (int \*)&len);

if (msgsock == -1)

perror("accept");

else

{

int idx = 0;

bool sucFlag = 0;

for ( idx = 0; idx < 10; idx++) {

if (st\_sock[idx].flag == 0) {

st\_sock[idx].clientSocket = msgsock;

st\_sock[idx].idx = idx;

st\_sock[idx].flag = 1;

sucFlag = 1;

break;

}

}

if (sucFlag == 0) {

closesocket(msgsock);

}

/\*显示已连接client的IP地址及端口\*/

struct sockaddr\_in \* Psockaddr\_in = (struct sockaddr\_in \*) &tcpaddr;

printf\_s("%s:%d,Welcome... !\n", inet\_ntoa(Psockaddr\_in->sin\_addr), ntohs(Psockaddr\_in->sin\_port));

client[inet\_ntoa(Psockaddr\_in->sin\_addr)] = TRUE;

HANDLE threadHandle = (HANDLE)\_beginthreadex(NULL, 0, workerThread, &st\_sock[idx], 0, &threadID);

}

} while (TRUE);

/\* 因为这个程序已经有了一个无限循环，所以套接字"sock"从来不显式关闭。然而，当进程被杀死或正常终止时，所有套接字都将自动地被关闭。\*/

closesocket(sock);

WSACleanup();

exit(0);

}

unsigned \_\_stdcall workerThread(void \*pSocket)

{

\_pstSock pMsgsock = (\_stSock \*)(pSocket);

int rval;

char rcv\_buf[1024],buf\_time[100];

sockaddr\_in \*Psockaddr\_in;

do {

/\*准备接收数据，阻塞\*/

if ((rval = recv(pMsgsock->clientSocket, rcv\_buf, 1024, 0)) < 0)

{

int errNO=WSAGetLastError();

perror("reading stream message");

break;

}

else if (rval == 0)

printf("ending connection \n");

else {

/\*向client回复当前时间\*/

time\_t tnow;

time(&tnow);

tm \*st\_tm = localtime(&tnow);

sprintf\_s(buf\_time, "%d-%d-%d:%d:%d:%d", st\_tm->tm\_year + 1900, st\_tm->tm\_mon + 1, st\_tm->tm\_mday, st\_tm->tm\_hour, st\_tm->tm\_min, st\_tm->tm\_sec);

printf("%s:idx %d-->::%s\n", buf\_time, pMsgsock->idx,/\*inet\_ntoa(Psockaddr\_in->sin\_addr),\*/ rcv\_buf);

if (send(pMsgsock->clientSocket, buf\_time, sizeof(buf\_time), 0) < 0)

perror("sending on stream socket");

}

} while (rval != 0);

pMsgsock->flag = 0;

closesocket(pMsgsock->clientSocket);

return 0;

}

// clientDemo.cpp : 此文件包含 "main" 函数。程序执行将在此处开始并结束。

//

#include "pch.h"

#include <iostream>

#pragma comment(lib,"WS2\_32.lib")

/\* File Name: client.c \*/

#include <winsock2.h>

#include <stdlib.h>

#include <stdio.h>

/\* 这个程序建立套接字，然后与命令行给出的套接字连接；连接结束时，在连接上发送

一个消息，然后关闭套接字。命令行的格式是：clientDemo 主机名 端口号

端口号要与服务器程序的端口号相同 \*/

int main(int argc, char \*argv[])

{

if (argc <=2) {

std::cout << "参数不足，请重新输入。。。";

return 0;

}

int sock;

struct sockaddr\_in server;

struct hostent \*hp;

char buf[1024] = "1234567";

char buf\_rev[1024];

WSADATA wsa;

//初始化环境

if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)

{

printf("fail to load socket.");

return 1;

}

sock = socket(AF\_INET, SOCK\_STREAM, 0);

if (sock < 0) {

perror("opening stream socket");

exit(1);

}

/\* 使用命令行中指定的名字连接套接字 \*/

server.sin\_family = AF\_INET;

printf\_s("准备连接主机 %s 的 %s端口\n", argv[1],argv[2]);

hp = gethostbyname(argv[1]);

if (hp == 0) {

fprintf(stderr, "%s: unknown host \n", argv[1]);

exit(2);

}

memcpy((char\*)&server.sin\_addr, (char\*)hp->h\_addr, hp->h\_length);

server.sin\_port = htons(atoi(argv[2]));

if (connect(sock, (struct sockaddr\*)&server, sizeof(server)) < 0) {

perror("connecting stream socket");

exit(3);

}

while (strcmp(buf, "exit") != 0)

{

scanf("%s", buf);

if (send(sock, buf, sizeof(buf), 0) < 0)

perror("sending on stream socket");

if (recv(sock, buf\_rev, 1024, 0) < 0)

{

perror("reading stream message");

break;

}

else

printf("-->%s\n", buf\_rev);

}

closesocket(sock);

return 0;

}

**实验结果分析**

首先启动服务器程序，接着启动client1

向服务器发送数据；启动client2，观察服务器是否能接受其连接，记录结果：

