# 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；

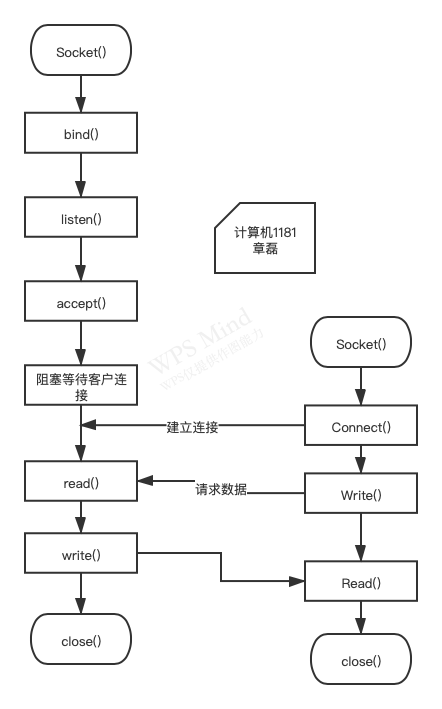
线路号：字符串： 10字节；

电压： 整型： 2字节；

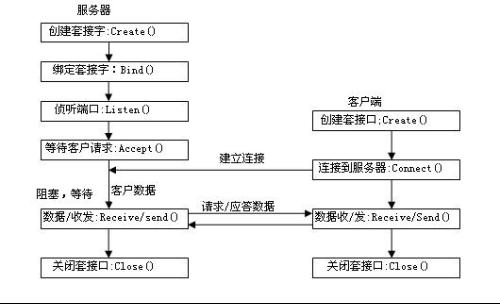
电流： 整型： 2字节}

## 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

答：要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，就要求服务器与客户端的交互要并发处理。我就想到用多线程来实现。启动多个线程，每个线程执行和一个客户端的交互。服务器端创建一个Socket,将他与Ip和设置的1234端口号绑定并设置监听模式等待连接请求，监听客户端的请求。当accept捕捉到一个连接时，就生成一个新的Socket。

相关代码如下：

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

{

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);

}

服务端相关代码如下：

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\*/

printf("ok\n");

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;

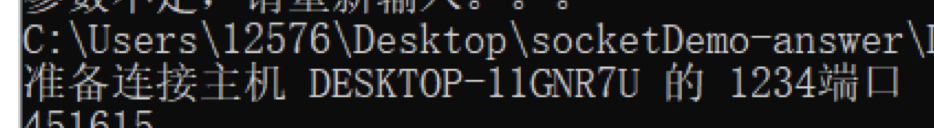
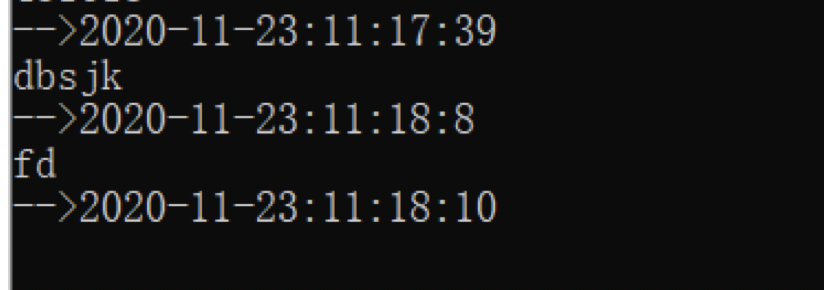
closesocket(pMsgsock->clientSocket);

return 0;

}

**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：