《网络程序设计》实践课

实验报告

实验名称：I/O 模型网络程序实验

实验类型：验证型

指导教师：贾浩

专业班级：

姓　　名：

学　　号：

实验评分

序号	实验分数构成	分数
1	实验结果(20 分)	
2	实验过程(40 分) （包括实验指导教师提问回答情况）	
3	实验报告书写规范（20 分）	
4	实验思考（20 分）	
实验总分		

一、实验目的

- 掌握 Winsock I/O 模型工作原理；
- 熟悉 I/O 模型中使用的 Winsock 接口函数；
- 掌握使用 I/O 模型进行网络程序设计的编程步骤；

二、实验设计

开发环境：Dev C++

1、在上述 I/O 模型中自选一个 I/O 模型，构建一个 TCP 服务器，该服务器能：

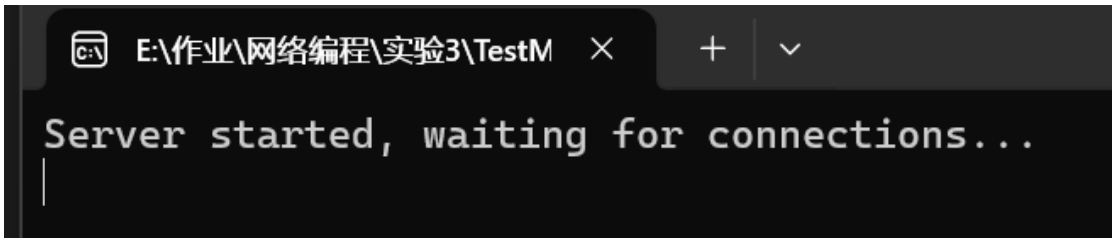
- 接受客户端连接时显示客户端的 IP，PORT 信息
- 接收客户端连接时显示其连接编号，客户端退出时显示关闭的连接编号
- 能显示客户端发来的数据
- 能从键盘输入数据并发到客户端
- 其他数据传送功能（可选）

2、 编写客户端程序，使之能：

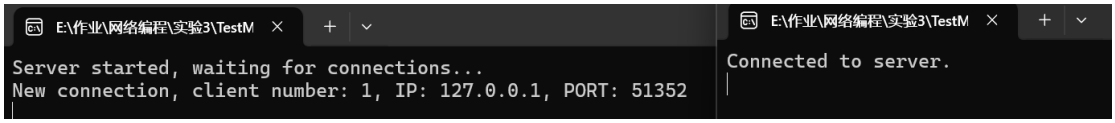
- 从键盘输入数据并发送到服务器
- 能接收服务器发来的数据
- 当输入“bye”时退出程序

三、实验过程（包含实验结果）

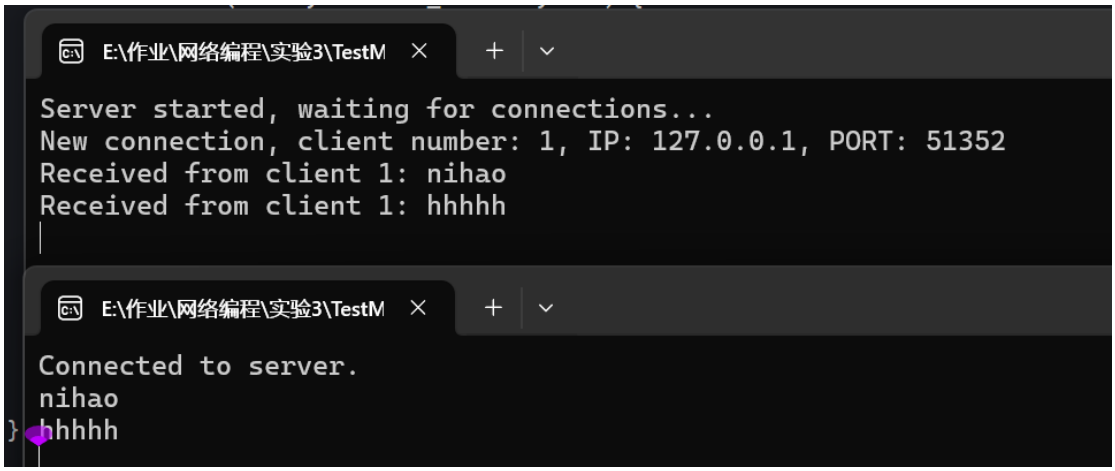
开启服务端：8884 端口等待客户端连接



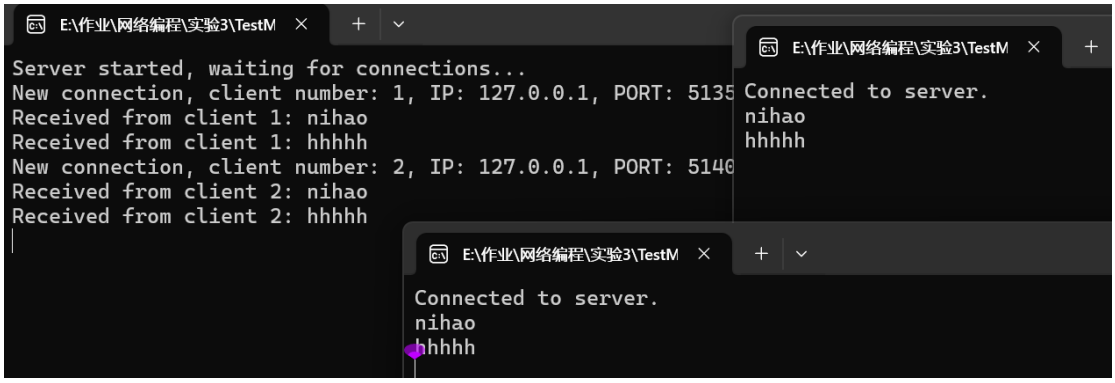
启动客户端，连接成功，显示客户端的 IP，PORT 信息和其连接编号



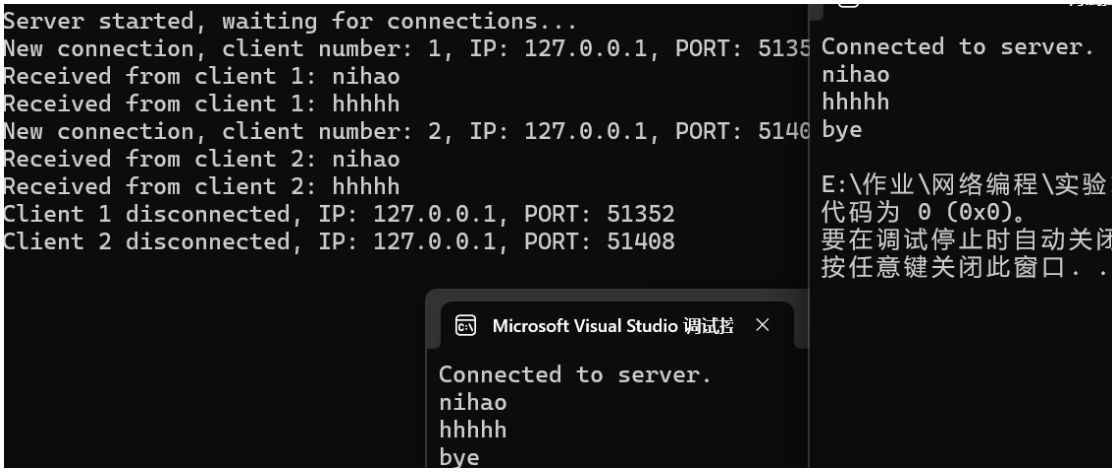
客户端发送消息，显示客户端的连接编号和消息内容



再添加一个客户端连接并发送数据

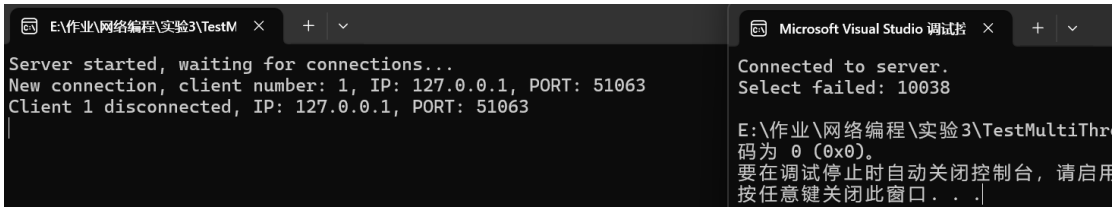


客户端退出，服务端处显示关闭的连接编号，客户端的 IP，PORT 信息。



四、讨论与分析（实验思考中问题理解）

问题 1：一开始客户端启动后与服务端建立连接后马上断开



解决方案：

输出“Select failed：10038”说明 select 函数调用失败，查看客户端的 select 函数发现错误地将标准输入（0）通过 FD_SET 宏添加到看 fd_set 集合中，而 select 函数仅支持对 Winsock 套接字描述符进行操作。因此调用 select 函数时，就会触发错误。

修改代码：建立 readfds 集合管理客户端套接字，每次循环 FD_ZERO 初始化 readfds 后，FD_SET 添加客户端套接字。

```
FD_ZERO(&readfds); // 4. 初始化 `readfds`
FD_SET(clientSocket, &readfds);
int activity = select(clientSocket + 1, &readfds, NULL, NULL, &timeout);
```

实验思考 1：你所选用的 I/O 模型是如何判断套接字上何时可以收发数据的或者数据收发已完成的？

我所选用的是 select 模型，该模型通过维护 fd_set 集合来判断套接字状态。调用 select 函数后，若套接字在可读集合（通过 FD_ISSET 检查）中，表明可以接收数据；若 recv 函数返回 0，则表示对方关闭连接，数据收发完成。

实验思考 2：简述你所使用的 I/O 模型的编程步骤。

Select 模型编程步骤：

1. 建立集合：定义 fd_set 集合 readfds 用于存放套接字。
2. 添加套接字：使用 FD_SET 将套接字（如监听套接字、客户端套接字）添加到集合 readfds。
3. 初始化集合：每次检查前，用 FD_ZERO 清空集合，再通过 FD_SET 重新添加待检查的套接字。
4. 调用 select 函数：指定检查的套接字范围、集合及超时时间，等待事件发生。
5. 处理返回结果：根据 select 返回值，用 FD_ISSET 检查套接字是否在集合中，判断事件类型（如新建连接、数据可读、客户端断开），并进行相应处理（如 accept 新连接、recv 接收数据、closesocket 关闭连接）。
6. 回到 3

实验思考 3：在你所使用的 I/O 中如何判断发生网络事件或者 IO 完成的套接字？

在 select 函数返回后，通过 FD_ISSET(sock, &readfds) 宏判断套接字 sock 是否在 readfds 集合中。

若在集合中：对于监听套接字，表明有新连接请求（调用 accept）。对于已连接的客户端套接字，recv 函数返回值非负时，表明有数据可读（可收发数据）；

若 recv 返回 0，则表示对方关闭连接，I/O 完成。

五、实验者自评（从实验设计、实验过程、对实验知识点的理解上给出客观公正的自我评价）

实验过程中，初期因误将标准输入加入 fd_set 导致客户端闪退，通过分析错误代码、查阅文档，修正后实现稳定连接，加深了对 select 模型仅支持套接字描述符的理解,明白了 select 模型通过维护 fd_set 集合、调用 FD_ISSET 宏判断网络事件的机制。整体达到实验目的，提升了网络编程与问题排查能力，对 Winsock I/O 模型有了扎实认知，代码实现符合需求。

六、 附录：关键代码（给出适当注释，可读性高）

服务端关键代码

```
SOCKET listenSocket, clientSockets[MAX_CLIENTS] = { 0 };
fd_set masterfds, readfds; // 1. 建立 `fd_set` 集合 `masterfds`
int maxSd, activity, i, valread;
int clientCount = 0; // 连接编号
listenSocket = socket(AF_INET, SOCK_STREAM, 0);
serverAddr.sin_family = AF_INET;
serverAddr.sin_addr.s_addr = INADDR_ANY;
serverAddr.sin_port = htons(8884);

FD_ZERO(&masterfds);
FD_SET(listenSocket, &masterfds); // 2. 将监听套接字添加到集合 `masterfds`
maxSd = listenSocket;
printf("Server started, waiting for connections...\n");

while (1) {
    readfds = masterfds; // 3. 确定检查集合 `readfds`
    FD_ZERO(&readfds); // 4. 初始化 `readfds`
    readfds = masterfds; // 5. 使用 `FD_SET` 逻辑上通过复制实现

    activity = select(maxSd + 1, &readfds, NULL, NULL, NULL); // 6. 调用 `select` 函数
    // 7. 根据返回值处理
    if (FD_ISSET(listenSocket, &readfds)) { // 新连接
        SOCKET newSocket = accept(listenSocket, (struct sockaddr*)&clientAddr, &clientAddrLen);
        if (newSocket == INVALID_SOCKET) {
            printf("Accept failed: %d\n", WSAGetLastError());
            continue;
        }
        for (i = 0; i < MAX_CLIENTS; i++) {
```

```
                    if (clientSockets[i] == 0) {
                        clientSockets[i] = newSocket;
                        clientCount++;
                        printf("New connection, client number: %d, IP: %s, PORT: %d\n",
                            clientCount, inet_ntoa(clientAddr.sin_addr),
ntohs(clientAddr.sin_port));
                        FD_SET(newSocket, &masterfds); // 处理时添加新套接字到
`masterfds`
                        if (newSocket > maxSd) maxSd = newSocket;
                        break;
                    }
                }
            }

            for (i = 0; i < MAX_CLIENTS; i++) {
                SOCKET sock = clientSockets[i];
                if (FD_ISSET(sock, &readfds)) { // 客户端数据
                    valread = recv(sock, buffer, BUFFER_SIZE, 0);
                    if (valread == 0) {
                        getpeername(sock, (struct sockaddr*)&clientAddr, &clientAddrLen);
                        // 查找连接编号并显示
                        for (int j = 0; j < MAX_CLIENTS; j++) {
                            if (clientSockets[j] == sock) {
                                printf("Client %d disconnected, IP: %s, PORT: %d\n",
                                    j + 1, inet_ntoa(clientAddr.sin_addr),
ntohs(clientAddr.sin_port));
                                break;
                            }
                        }
                        closesocket(sock);
                        FD_CLR(sock, &masterfds);
                        clientSockets[i] = 0;
                    }
                    else {
                        buffer[valread] = '\0';
                        printf("Received from client %d: %s\n", i + 1, buffer);
                    }
                }
            }

            // 处理键盘输入并发送给所有客户端
            if (_kbhit()) {
                fgets(buffer, BUFFER_SIZE, stdin);
                buffer[strcspn(buffer, "\n")] = 0;
```

```
            for (i = 0; i < MAX_CLIENTS; i++) {
                SOCKET sock = clientSockets[i];
                if (sock != 0) send(sock, buffer, strlen(buffer), 0);
            }
        }
    }
```

客户端关键代码

```
    SOCKET clientSocket;
    struct sockaddr_in serverAddr;
    clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    serverAddr.sin_port = htons(8884);

    fd_set readfds; // 1. 建立 `fd_set` 集合 `readfds`
    struct timeval timeout = { 0, 100000 };

    while (1) {
        FD_ZERO(&readfds); // 4. 初始化 `readfds`
        FD_SET(clientSocket, &readfds); // 2. 将客户端套接字添加到集合 `readfds`（5. 使用 `FD_SET`）

        int activity = select(clientSocket + 1, &readfds, NULL, NULL, &timeout); // 6. 调用 `select` 函数
        // 7. 根据返回值处理
        if (FD_ISSET(clientSocket, &readfds)) { // 接收服务器数据
            int valread = recv(clientSocket, buffer, BUFFER_SIZE, 0);
            if (valread == 0) {
                printf("Server disconnected.\n");
                break;
            }
            buffer[valread] = '\0';
            printf("Received from server: %s\n", buffer);
        }

        // 处理键盘输入
        if (_kbhit()) {
            fgets(buffer, BUFFER_SIZE, stdin);
            buffer[strcspn(buffer, "\n")] = 0;
            if (strcmp(buffer, "bye") == 0) break; // 输入 `bye` 退出
            send(clientSocket, buffer, strlen(buffer), 0);
        }
    }
```