第一章 理解网络编程和套接字

网络编程就是编写程序使两台联网的计算机相互交换数据。
套接字是网络数据传输用的软件设备。

服务端套接字（监听套接字）

创建套接字（相当于买一个电话机）

```
#include <sys/socket.h>
int socket(int domain, int type, int protocol);
//成功时返回一个文件描述符，失败时返回-1
```

为套接字分配地址 ip（地址）和port（端口号）(类似于为电话机分配号码)

```
#include <sys/socket.h>
int bind(int sockfd, struct sockaddr* myaddr, socklen_t addrlen);
//成功时返回0，失败时返回-1
```

将套接字转化为可接收的状态

```
#include <sys/socket.h>
int listen(int sockfd, int backlog)
//成功时返回0，失败时返回-1
```

如果有人为了完成数据传输请求连接，就需要调用以下函数进行受理：

```
#include <sys/socket.h>
int accept(int socked, struct sockaddr* addr, socklen_t* addrlen);
//成功时，返回文件描述符，失败时返回-1
```

客户端构建

客户端只有"调用socket函数创建套接字"和"调用connect函数向服务端发送连接请求"这两个步骤

```
#include <sys/socket.h>
int connect(int sockfd, struct sockaddr* serv_addr, socklen_t addrlen);
```

文件描述符

在Linux中，文件描述符（文件句柄）用一个int类型表示，创建套接字成功后也会返回一个文件描述符；默认情况下标准输入Standard Input的文件描述符为0、标准输出Standard Output的文件描述符为1、标准错误Standard Error的文件描述符为2；用户自己创建的文件描述符则从3开始；不管是文件描述符（Linux）还是句柄（Windows）都是操作系统创建的文件或套接字而赋予的数而已

打开文件

```
#include <fcntl.h>
int open(const char* file_path, int flag); //file_path是文件的路径、flag是打开模式信息
//成功创建时返回一个文件描述符，否则返回-1
```

表1-2　文件打开模式

打开模式	含　义
O_CREAT	必要时创建文件
O_TRUNC	删除全部现有数据
O_APPEND	维持现有数据，保存到其后面
O_RDONLY	只读打开
O_WRONLY	只写打开
O_RDWR	读写打开

表1-2是open函数第二个参数flag可能的常量值及含义。如需传递多个参数，则　应通过位(&)运算或者或(|)运算符组合并传递

关闭文件

```
#include <unistd.h>
int close(int fd); //fd是需关闭的文件描述符
//成功时返回0，否则返回-1
```

将数据写入文件

```
#include <unistd.h>
ssize_t write(int fd, const void* buf, size_t nbytes);//fd是显示数据传输对象的文件描述符; buf是保存要传输数据的缓冲地址值; nbytes是要传输数据的字节数
//成功时返回写入的字节数，失败时返回-1
//ssize_t是有符号整型（signed int）
//size_t是无符号整型（unsigned int）
```

读取文件中的数据

```
#include <unistd.h>
ssize_t read(int fd, void* buf, size_t count);//fd是显示数据接收对象的文件描述符; buf是保存要传输数据的缓冲地址值; nbytes是要传输数据的字节数
```

基于Windows平台的实现

```
#include <winsock2.h>
int WSAStartup(WORD wVersionRequested, LPWSADATA lpWSAData);//第一个参数是Winsock版本信息; 第二个是WSADATA结构体变量的地址值
//成功时返回0，失败时返回非零的错误代码值

int WSACleanup(void);//关闭socket
//成功时返回0，失败时返回SOCKET_ERROR
```

```
#include <winsock2.h>
SOCKET socket(int af, int type, int protocol);
//成功时返回套接字句柄，失败时返回INVALID_SOCKET
```

```
#include <winsock2.h>
int bind(SOCKET s, const struct sockaddr *name, int namelen);
//成功时返回0,失败时返回SOCKET_ERROR
```

```
#include <winsock2.h>
int listen(SOCKET s, int backlog)//backlog表示最大等待连接队列的长度(即操作系统允许的未处理连接的数量)；超出此数量的连接请求可能会被拒绝或挂起
//成功时返回0，失败时返回SOCKET_ERROR
```

```
#include <winsock2.h>
SOCKET accept(SOCKET s, struct sockaddr *addr, int *addrlen)
//与Linux的accept函数相同，调用其受理客户端连接请求
//成功时返回套接字句柄，失败时返回INVALID_SOCKET
```

```
#include <winsock2.h>
int connect(SOCKET s, const struct sockaddr *name, int namelen);
//与Linux的connect函数相同，调用其从客户端发送连接请求
//成功时返回0，失败时返回SOCKET_ERROR
```

```
#include <winsock2.h>
//Windows中专门有个函数来关闭套接字的函数
int closesocket(SOCKET s);
//成功时返回0，失败时返回SOCKET_ERROR
```

Winsock库与Linux下的socket库

在Windows中，需要初始化Winsock库和注销相关代码、数据类型信息外，其余部分与Linux环境下的示例并无区别

第一章习题

1、套接字在网络编程中的作用是什么？为何称它为套接字?

答：套接字在网络中起到数据传输的作用，它屏蔽了底层协议和硬件实现的复杂性，为开发者提供简单而一致的接口。"套接字"（Socket）的名称来源于其形象的比喻，指的是“插座”的功能。在现实中，一个插座和插头通过插接在一起实现连接和电流传输。同样，在网络编程中：一个套接字表示通信的一端。当两个套接字成功“插接”（即建立连接）后，它们之间就可以进行数据的传输。

2、在服务端创建套接字后，会依次调用listen函数和accept函数。请比较并说明二者的作用。

答：listen是服务端开始准备接受连接的第一步，它将套接字置于侦听模式，类似于告诉操作系统“我准备好连接了”；accept是服务端具体处理连接请求的操作，它从队列中取出一个连接并生成用于通信的新套接字，类似于说“好，现在拿出一个连接来处理吧！”

比较 `listen` 和 `accept` 的作用

特性	`listen`	`accept`
阶段	用于准备接受连接	用于处理具体的连接
功能	将套接字设置为侦听模式，等待客户端连接请求	接受一个客户端连接并生成专用通信套接字
影响的套接字	作用于侦听套接字	生成新的通信套接字
阻塞性	不阻塞	默认情况下阻塞，直到有连接到达
返回值	无返回值	返回一个新的套接字描述符
调用条件	必须在 `bind` 之后调用	必须在 `listen` 之后调用

3、Linux中，对套接字数据进行I/O时可以直接使用文件I/O相关函数；而在Windows中则不可以，原因为何?

答：Linux 的“统一 I/O 模型”使得套接字可以直接使用文件 I/O 函数，而 Windows 中文件和套接字的实现机制不同，因此必须使用各自专属的 API。两种设计各有优劣，Linux 注重统一性，Windows 更注重功能模块化和优化专用性。

特性	Linux	Windows
套接字与文件的关系	套接字是文件的一种	套接字和文件是完全不同的资源类型
I/O 函数的统一性	套接字可以直接使用文件 I/O 函数	套接字需要使用 Winsock 提供的专用函数
底层设计	统一的 I/O 模型，通过文件描述符进行资源管理	分离的模型，文件和套接字通过不同的句柄管理
开发者体验	操作简单，I/O 函数复用性高	操作复杂，需要根据资源类型选择不同的接口

4、创建套接字后一般会给它分配地址，为什么?

答：创建套接字后会调用bind函数给套接字分配唯一的ip和端口号地址，目的是为了让客户端能够通过该地址和端口号来准确的找到服务端的特定进程，从而实现网络的数据传输。

5、Linux中的文件描述符与Windows的句柄实际上非常类似。请以套接字为对象说明他们的含义。

答：无论是Linux中的文件描述符还是Windows中的句柄，都是用来抽象资源访问的工具。对于套接字，两者的核心概念是相同的——提供一个接口，让程序能够通过这个标识符与内核管理的套接字资源交互。不同之处在于它们的实现细节和设计哲学：Linux倾向于使用统一的文件抽象，而Windows则更倾向于专门化的对象管理。

6、底层文件I/O函数与ANSI标准定义的文件I/O函数之间有何区别?

答：底层文件I/O函数：适用于需要细粒度控制文件操作的场景，如操作设备文件、内存映射文件、大文件分块处理等。

ANSI标准文件I/O函数：适用于一般的文件处理任务，如读取配置文件、写入日志文件、处理文本文件等。