TCP/IP网络编程(二)

笔记本: 网络编程
创建时间: 2018/10/30 18:53 **更新时间:** 2018/10/31 20:08
作者: xiangkang94@outlook.com
标签: 第二章(套接字类型与协议设置)

---

1. 创建套接字的函数原型

```
#include <sys/socket.h>
int socket(int domain, int type, int protocol);
//domian：协议族；Type：套接字类型；protocol：协议信息
```

1. 协议族
    - PF_INET IPv4
    - PF_INET6 IPv6
    - PF_LOCAL 本地通信的UNIX协议族
    - PF_PACKET 底层套接字协议族
    - PF_IPX IPX Novell协议族
2. 套接字类型
    - 面向连接的套接字(SOCK_STREAM)
        - 可靠
        - 按序
        - 没有数据边界(分批传递，打包接受，缓冲池技术)
    - 面向消息的套接字(SOCK_DGRAM)
        - 快速
        - 无序
        - 有数据边界(多少次传输则对于多少次接收)
        - 限制每次传输数据大小
3. 协议的最终选择
    - 由于前两个参数基本可以确定具体的协议，通常第三个参数可以指定为0，除非遇到“同一协议族中存在多个数据传输方式相同的协议”
4. Q&A
    1. tcp_server.c和tcp_client.c中需要多次调用read函数读取服务器调用1次write函数传递的字符串。更改程序，使服务器端多次调用（次数自拟）write函数传输数据，客户端调用1次read函数进行读取。为达到这一目的，客户端需延迟调用read函数，因为客户端要等待服务器端传输所有数据。windows和linux都通过下列代码延迟read或recv函数的调用。
    for(i=0;i<3000;i++)
        printf("wait time %d \n",i);

让CPU执行多余任务以延迟代码运行的方式称为“Busy waiting”。使用得当即可推迟函数调用。

```
#include <stdio.h>
#include <stdlib.h>
#include <WinSock2.h>
void ErrorHandling(char *message) {
    fputs(message, stderr);
    fputs("\n", stderr);
    exit(1);
}
int main(int argc, char * argv[]) {
    WSADATA wsaData;
    SOCKET hServSock, hClntSock;
    SOCKADDR_IN servAddr, clntAddr;
    int szClntAddr;
    char message[] = "Hello World!";
    if (argc != 2) {
        printf("Usage : %s <port>\n", argv[0]);
```

```
            exit(1);
        }
        if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
            ErrorHandling("WSAStartup() error!");
        hServSock = socket(PF_INET, SOCK_STREAM, 0);
        if (hServSock == INVALID_SOCKET)
        ErrorHandling("socket() error!");
        memset(&servAddr, 0, sizeof(servAddr));
        servAddr.sin_family = AF_INET;
        servAddr.sin_addr.s_addr = htonl(INADDR_ANY);
        servAddr.sin_port = htons(atoi(argv[1]));
        if (bind(hServSock, (SOCKADDR *)&servAddr, sizeof(servAddr)) == SOCKET_ERROR)
            ErrorHandling("bind() error!");
        if (listen(hServSock, 5) == SOCKET_ERROR)
            ErrorHandling("listen() error!");
        szClntAddr = sizeof(clntAddr);
        hClntSock = accept(hServSock, (SOCKADDR*)&clntAddr, &szClntAddr);
        if (hClntSock == INVALID_SOCKET)
            ErrorHandling("accept() error!");
        //fputs("sending message...\n",stderr);
        for (int i = 0; i < strlen(message); i++) {
            send(hClntSock, &message[i], sizeof(message[i]), 0);
            printf("sending %c\n", message[i]);
        }
        send(hClntSock, &message[strlen(message)], sizeof(message[strlen(message)]), 0);
        closesocket(hClntSock);
        closesocket(hServSock);
        WSACleanup();
        return 0;
    }
```