【实验题目】文件传输实验（选做）

【实验目的】学习利用套接字传送文件

【实验内容】

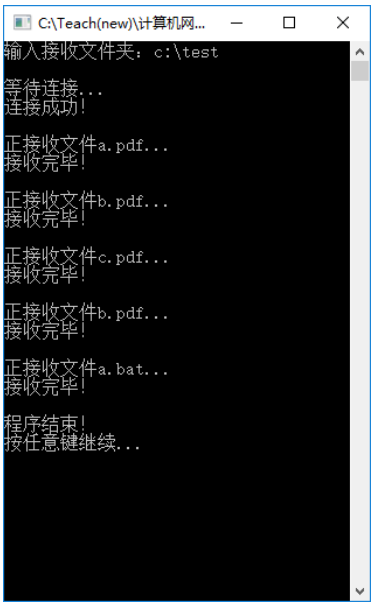
利用数据表示实验和 Echo 实验实现以下功能：

（1） 运行服务器端程序，输入接收文件的文件夹，然后等待客户端连接，并接收客户端发来的文件并保存，直到客户端关闭连接。有重名文件时，文件名增加序号保存。

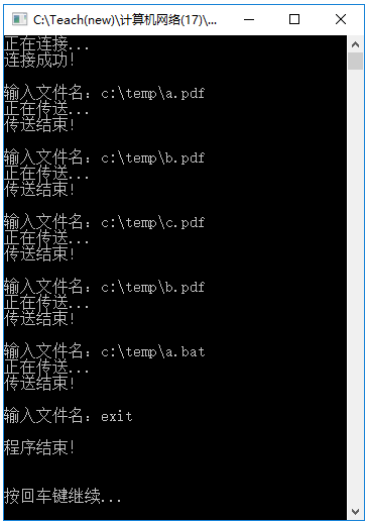
（2） 客户端先连接服务器，每次输入一个文件名（包含路径）就传输它，直到输入 exit 时退出并关闭套接字。

【参考运行截屏】

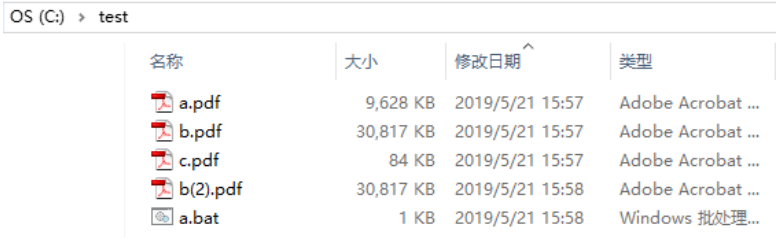
服务器：



客户端：

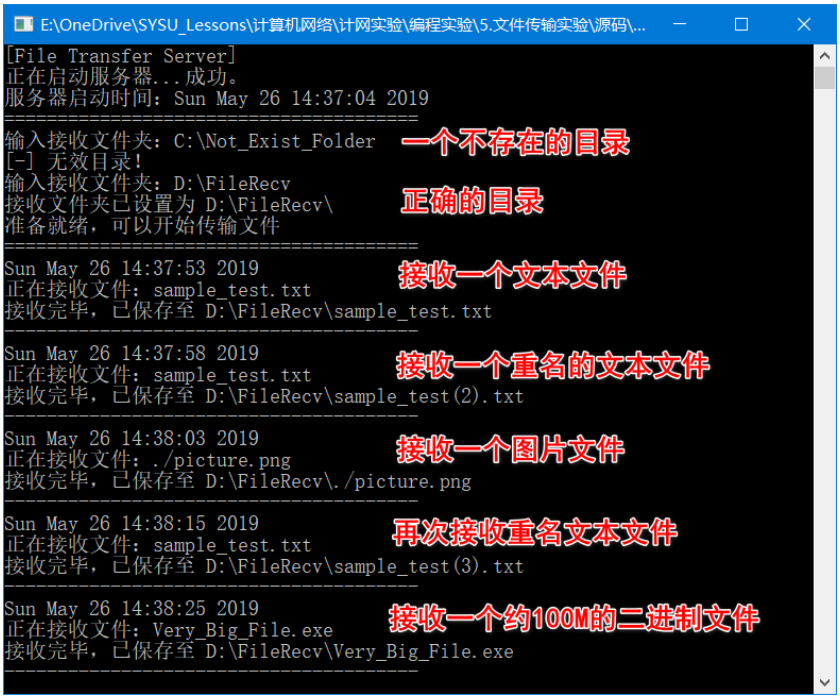


目的文件夹：

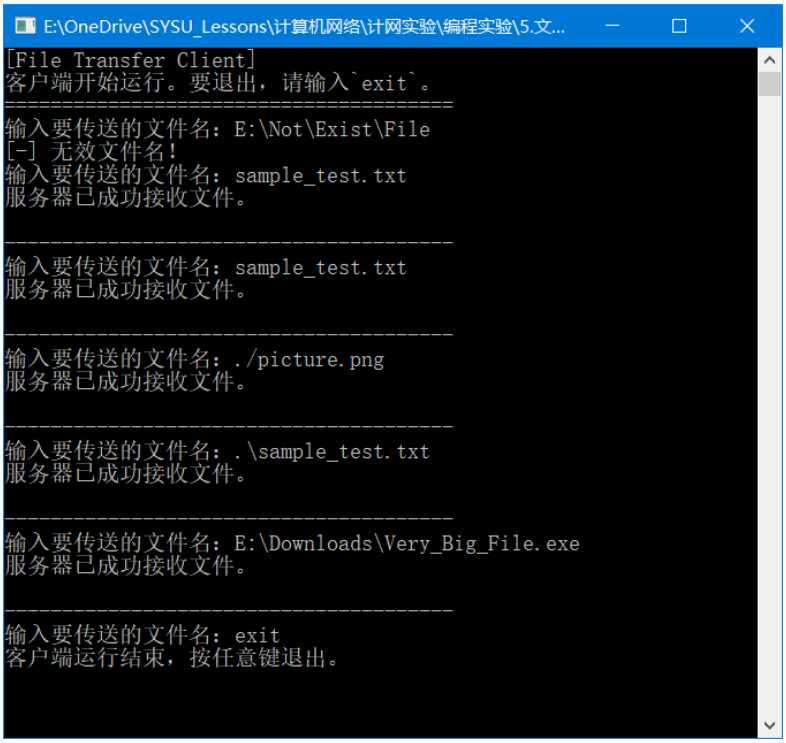


【实验结果】

运行成功后，服务器截屏：



客户端运行截屏：



目标文件夹：

接收的所有文件均可打开：文本文件可以阅读，图片文件可以浏览，可执行文件可以正确运行。

服务器程序：

```c
#include <stdio.h>
#include <time.h>
#include <winsock2.h>
#include <conio.h>
#include <string>
#include <io.h>
#pragma comment(lib, "ws2_32.lib") // 使用winsock 2.2 library
#define BUFLEN 2000                // 缓冲区大小
using namespace std;

void validateFilename(const char* folder, char* filename);

int main()
{
    printf("[File Transfer Server]\n 正在启动服务器...");

    struct sockaddr_in clsin; // the from address of a client
    struct sockaddr_in sin;   // an Internet endpoint address
    SOCKET msock, ssock;      // master & slave sockets
    u_short port = 50500;     // server port

    WSADATA wsadata;
    WSAStartup(MAKEWORD(2, 0), &wsadata); // 加载Winsock library，使用2.0 版本

    msock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP); // 创建主socket
    memset(&sin, 0, sizeof(sin));                      // 清零
    sin.sin_family = AF_INET;                          // 因特网地址簇 (INET-Internet)
    sin.sin_addr.s_addr = INADDR_ANY;                  // 监听所有(接口的)IP 地址
    sin.sin_port = htons(port);                        // 监听的端口号（网络序）
    bind(msock, (struct sockaddr *)&sin, sizeof(sin)); // 绑定监听的IP 地址和端口号
    listen(msock, 5);                                  // 建立长度为5 的连接请求队列，并把到来的连接请求加入队列等待处理

    time_t now = time(NULL);
    printf("成功。\n 服务器启动时间: %s", ctime(&now));
    printf("=======================================\n");

    char folder[100] = {0};
    do {
        printf("输入接收文件夹: ");
        scanf("%s", folder);
        if(access(folder, F_OK) != 0) {  // 检查目录是否存在且有访问权限
            fprintf(stderr, "[-] 无效目录! \n");
        }
    } while(access(folder, F_OK) != 0);
    if(folder[strlen(folder)-1] != '\\') {
        folder[strlen(folder)+1] = '\0';
```

```
        folder[strlen(folder)] = '\\';  // 保证目录以反斜杠结尾
    }
    printf("接收文件夹已设置为 %s\n 准备就绪，可以开始传输文件\n", folder);
    printf("========================================\n");

    char buf[BUFLEN + 1]; // 建立缓冲区

    while (!_kbhit())
    {                                                    // 检测是否
有按键,如果没有则进入循环体执行
        int alen = sizeof(struct sockaddr);              // 取到地址
结构的长度
        ssock = accept(msock, (struct sockaddr *)&clsin, &alen); // 从 socket,
阻塞地从连接请求队列中创建连接

        char filename[100] = {0};
        int recvlen = recv(ssock, filename, 100, 0); // 接收信息
        buf[recvlen] = '\0';                         // 保证以空字符结尾

        now = time(NULL);
        char *timestr = ctime(&now);
        printf("%s", timestr);
        printf("正在接收文件: %s\n", filename);

        validateFilename(folder, filename); // 为重名文件添加序号

        string full_filename = (string(folder) + filename);
        FILE* fp = fopen(full_filename.c_str(), "wb");
        int length = 0;
        /* 循环接收数据并写入文件 */
        while((length = recv(ssock, buf, BUFLEN, 0)) > 0) {
            fwrite(buf, sizeof(char), length, fp);
        }
        fclose(fp);

        printf("接收完毕，已保存至 %s\n", full_filename.c_str());
        printf("----------------------------------------\n");

        strncpy(buf, "服务器已成功接收文件。\n", BUFLEN);
        send(ssock, buf, strlen(buf) + 1, 0); // 发送信息
        shutdown(ssock, SD_SEND);             // 禁止收发数据
        closesocket(ssock);                   // 关闭从 socket
    }

    closesocket(msock); // 关闭主 socket
    WSACleanup();       // 卸载 winsock library
    printf("服务器已关闭，按任意键退出。\n");
    getchar();
    return 0;
}

/* 构造有效的文件名（即为重名文件添加序号） */
void validateFilename(const char* folder, char* filename)
{
    string full_filename = (string(folder) + filename);
    char temp[300];
    char numstr[10];
    int num = 1;
    while(access(full_filename.c_str(), 0)==0) {    // 文件已存在（重名）
        int dotpos = strrchr(filename, '.') - filename; // 从右往左找到小数点
的位置
        if(strrchr(filename, '(')!=NULL && strrchr(filename, ')')!=NULL) {
            int leftpos = strrchr(filename, '(') - filename;    // 左括号的位
置
            strncpy(temp, filename+dotpos, 300);
```

```
            filename[leftpos] = '\0';
            strncat(filename, temp, 300);   // 恢复不带序号的文件名
        }
        dotpos = strrchr(filename, '.') - filename; // 更新小数点的位置
        num++;                              // 递增序号
        numstr[0] = '(';                    // 添加左括号
        itoa(num, numstr+1, 10);            // 添加序号
        numstr[strlen(numstr)+1] = '\0';
        numstr[strlen(numstr)] = ')';   // 添加右括号
        strncpy(temp, filename, dotpos);
        temp[dotpos] = '\0';
        strcat(temp, numstr);
        strcat(temp, filename + dotpos);
        strncpy(filename, temp, 300);   // 把添加了序号的文件名拷贝回filename

        full_filename = (string(folder) + filename);
    }
}
```

客户端程序：

```
#include <stdio.h>
#include <winsock2.h>
#include <io.h>
#include <string>
#define BUFLEN 2000                 // 缓冲区大小
#pragma comment(lib, "ws2_32.lib") // 使用Winsock 2.0 Library
using namespace std;

int connectAndSend();

int main()
{
    printf("[File Transfer Client]\n客户端开始运行。要退出，请输入`exit`。\n");
    printf("=======================================\n");

    while (connectAndSend());

    printf("客户端运行结束，按任意键退出。\n");
    getchar();
    return 0;
}

/* ***** connectAndSend()函数说明 ********
 * 该函数执行完整的建立连接和断开连接过程，
 * 写在一个函数中的好处是可以仅运行一次程序
 * 就可以多次接收文件。
 *
 * 本函数的返回值：若用户输入了exit或遇到错误
 * 则返回0；其他情况均返回1。
 */
int connectAndSend()
{
    char *host = "127.0.0.1";
    u_short serverport = 50500;
    struct sockaddr_in sin;
    char buf[BUFLEN + 1];
    SOCKET sock;

    WSADATA wsadata;
    WSAStartup(MAKEWORD(2, 0), &wsadata);

    sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
    memset(&sin, 0, sizeof(sin));
    sin.sin_family = AF_INET;
```

```
    sin.sin_addr.s_addr = inet_addr(host);
    sin.sin_port = htons(serverport);
    int ret = connect(sock, (struct sockaddr *)&sin, sizeof(sin));
    if (ret == -1)
    {
        printf("[-] 连接到服务器失败! \n");
        closesocket(sock);
        WSACleanup();
        return 0;
    }

    char filename[128] = {0};
    do {
        printf("输入要传送的文件名: ");
        fgets(filename, BUFLEN, stdin);
        filename[strlen(filename) - 1] = '\0'; // 把换行符替换为空字符
        if (strcmp("exit", filename) == 0)
        {
            closesocket(sock);
            WSACleanup();
            return 0;
        } // 用户退出
        if(access(filename, R_OK) != 0) {
            fprintf(stderr, "[-] 无效文件名! \n");
        }
    } while(access(filename, R_OK) != 0);

    size_t filename_lastslash = string(filename).find_last_of('\\');  // 从完整路径中
文件名
    if(filename_lastslash == string::npos) {
        send(sock, filename, strlen(filename) + 1, 0);    // 发送文件名
    }
    else {
        string filename_partial = string(filename).substr(filename_lastslash+1);
        send(sock, filename_partial.c_str(), strlen(filename) + 1, 0);  // 不带路径的
文件名
    }


    FILE* fp = fopen(filename, "rb");
    int length;
    /* 循环读取文件并发送 */
    while((length = (int)fread(buf, sizeof(char), BUFLEN, fp)) > 0) {
        send(sock, buf, length, 0);
    }
    fclose(fp);
    shutdown(sock, SD_SEND);  // 发送FIN 包，表示发送完毕

    int recvlen = recv(sock, buf, BUFLEN, 0); // 接收反馈信息
    if (recvlen == SOCKET_ERROR)
    {
        printf("[-] Error: %ld.\n", GetLastError());
    }
    else if (recvlen == 0)
    {
        printf("[-] Server closed!");
    }
    else if (recvlen > 0)
    {
        buf[recvlen] = '\0';
        printf("%s\n", buf);
        printf("----------------------------------------\n");
    }
    closesocket(sock);
    WSACleanup();
```

```
    return 1;
}
```

【实验体会】

本次编程实验综合运用了前面的“数据表示实验”、“Echo 实验”、“应用层实验”中的内容，实现了一套通过 TCP 协议传输文件的 C/S 程序。为了使程序更健壮，我增加了许多检查用户输入、检查文件是否存在之类的代码，使得程序能够处理各种意外情况，不过也可能还有其他意外情况尚未考虑。对于接收到的重复名称的文件，我的程序可以自动为其添加序号。如果接收到很多个重复的文件，它们则会拥有递增的序号，不会发生文件覆盖的情况。还有路径处理，即使用户输入的目录不以“\”结尾，程序也能正常处理路径，使得文件路径永远是有效的。

在编写程序的过程中，我对 Socket 的 send 和 recv 有了更深的理解，并学会了如何把 send、recv 与 fread、fwrite 结合起来实现通过网络进行文件读写，还学会了如何通过有限缓冲区进行文件的循环读写，使得能够分批收发体积很大的文件。

要想传输所有类型的文件而不仅仅是文本文件，需要使用二进制方式打开文件，也就是`rb`和`wb`。文本文件也视作二进制文件。这是一个容易忽略的细节，如果只是以文本方式打开文件的话，那么在处理二进制文件时就可能发生意外情况。

尽管我的源文件是 cpp 格式，不过我的代码主要由 C 语言构成。字符串 I/O 使用的是 printf、fprintf、scanf、gets 等函数，文件操作使用的是 fopen、fread、fwrite 等函数。唯一使用的 C++特性是 string 类，string 类可以很方便地进行字符串处理，如拼接、取子串、查找字符等，比 C 风格字符串更加易于使用，而且不容易出错。使用 string 类处理字符串后用 c_str()方法转换为 C 风格字符串，就能被其他 C 函数处理了，兼容性很好。

最后说明一点，我使用 gcc 编译器编译服务器和客户端程序，需要手动添加“-lws2_32”编译参数以支持 Windows 的 Socket 接口。