

网络服务器综合配置+TCP 端口扫描

一、目的

1. 网络服务器综合配置，加深对 TCP/IP 协议的理解。明确 IP 地址、域名的实质。
2. 能够完成对 DNS、WEB、FTP 端口的配置并成功通过计算机访问。

二、过程要求

配置 PC1、PC2 和 PC3 三台计算机网络信息，以 PC1 为例：IP 地址：192.168.0. x （x 为实验室 PC1 的编号）。子网掩码：255.255.255.0。默认网关：空。DNS1 服务器 IP 地址：PC3 的 IP 地址。DNS2 服务器 IP 地址：空。

检查局域网的连通性，确保 PC1、PC2 和 PC3 之间可以 ping 通。配置 DNS 服务器，建立两个域名与 IP 地址之间的对应关系。检查 DNS 配置的正确性，在 PC1 的 DOS 命令窗口下，分别输入 dos>ping www.abc-x.com 和 dos>ping ftp. abc-x. com。 若可以 ping 通，则说明 DNS 服务器配置正确。

配置 Web 服务器，端口号为 80，IP 地址= PC2 的 IP 地址。在 c:\test\目录下建立 index.html 网页，并发布网页。检查 Web 服务器配置的正确性，在 PC1 的浏览器地址栏中输入 www.abc-x.com。若测试网页在浏览器上可以正确显示，则说明 Web 服务器配置正确。

配置 FTP 服务器，在 FTP 服务器的 c 盘根目录下建立目录：c:\子目录，并在子目录下均复制一些文件。检查 FTP 服务器配置的正确性，在 PC1 的浏览器地址栏输入 ftp. abc-x. com。若 PC1 的浏览器上可以显示 c 盘下两个子目录及其相关文件信息，则认为 FTP 服务器配置正确。

三、相关知识

DNS:

域名系统(Domain Name System,DNS)是 Internet 上解决网上机器命名的一种系统。就像拜访朋友要先知道别人家怎么走一样，Internet 上当一台主机要访问另外一台主机时，必须首先获知其地址，TCP/IP 中的 IP 地址是由四段以 “.” 分开的数字组成(此处以 IPv4 的

地址为例, IPv6 的地址同理), 记起来总是不如名字那么方便, 所以, 就采用了域名系统来管理名字和 IP 的对应关系。

虽然因特网上的节点都可以用 IP 地址惟一标识, 并且可以通过 IP 地址被访问, 但即使是将 32 位的二进制 IP 地址写成 4 个 0~255 的十位数形式, 也依然太长、太难记。因此, 人们发明了域名(Domain Name), 域名可将一个 IP 地址关联到一组有意义的字符上去。用户访问一个网站的时候, 既可以输入该网站的 IP 地址, 也可以输入其域名, 对访问而言, 两者是等价的。例如: 微软公司的 Web 服务器的 IP 地址是 207.46.230.229, 其对应的域名是 `www.microsoft.com`, 不管用户在浏览器中输入的是 207.46.230.229 还是 `www.microsoft.com`, 都可以访问其 Web 网站。

Web:

web (World Wide Web) 即全球广域网, 也称为万维网, 它是一种基于超文本和 HTTP 的、全球性的、动态交互的、跨平台的分布式图形信息系统。是建立在 Internet 上的一种网络服务, 为浏览者在 Internet 上查找和浏览信息提供了图形化的、易于访问的直观界面, 其中的文档及超级链接将 Internet 上的信息节点组织成一个互为关联的网状结构。

FTP:

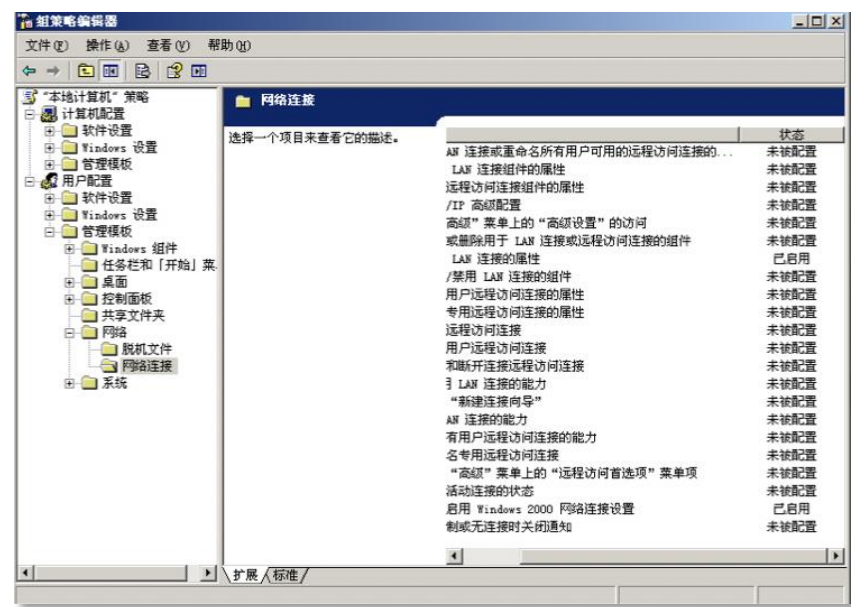
文件传输协议 (File Transfer Protocol, FTP) 是用于在网络上进行文件传输的一套标准协议, 它工作在 OSI 模型的第七层, TCP 模型的第四层, 即应用层, 使用 TCP 传输而不是 UDP, 客户在和服务器建立连接前要经过一个“三次握手”的过程, 保证客户与服务器之间的连接是可靠的, 而且是面向连接, 为数据传输提供可靠保证。

FTP 允许用户以文件操作的方式 (如文件的增、删、改、查、传送等) 与另一主机相互通信。然而, 用户并不真正登录到自己想要存取的计算机上面而成为完全用户, 可用 FTP 程序访问远程资源, 实现用户往返传输文件、目录管理以及访问电子邮件等等, 即使双方计算机可能配有不同的操作系统和文件存储方式。

四、实现原理及流程图

实验内容一:

在实验开始之前首先进行网卡的关闭禁用网卡“本地连接”, 实验中只使用“本地连接”, 以避免干扰。在这个地方进行关闭。(注意关闭的时候一定要先配置好为网卡绑定 IP 地址, 否则会出现无法设置的情况):

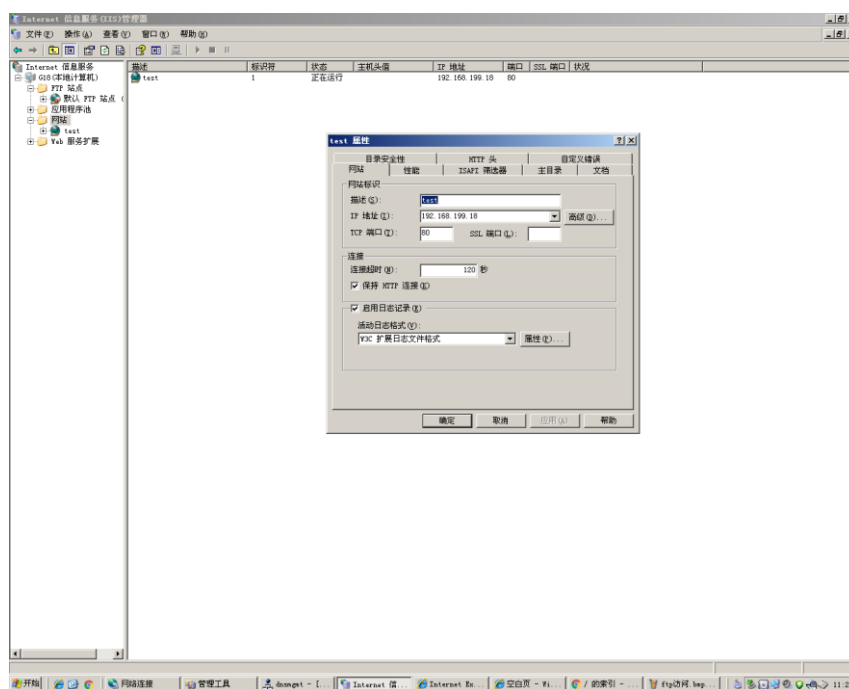


之后是 DNS 服务器的创建及连接，本次实验采取三个人一个小组的形式（我们组 IP 地址最后两位分别为 17、18、19），一个小组中需要分别有人完成 DNS 服务器、Web 服务器以及 FTP 服务器，想要相互连接并减少冲突，需要让三台主机在同一局域网下，因此需要更改本机的 ip 地址此 IP 同时也将成为本机的默认 IP 地址。选“使用下面的 IP 地址”，在“IP 地址”一栏输入“192.168.19.17（该数字为本机的 IP 地址）”；子网掩码一栏 输入“255.255.255.0”；如果本机同时又是本网内的服务器，则“默认 网关”和“首选 DNS 服务器”两栏也均填入此默认 IP 地址；如果本机不是本网 内的服务器，则一般“默认网关”和“备用 DNS 服务器”两栏的值为服务器的 IP 地址，而“首选 DNS 服务器”仍然为本机的默认 IP 地址。

配置完成之后进行 DNS 服务器以及 FTP 服务器的配置。打开 DNS 控制台，选择主要区域，然后下一步：

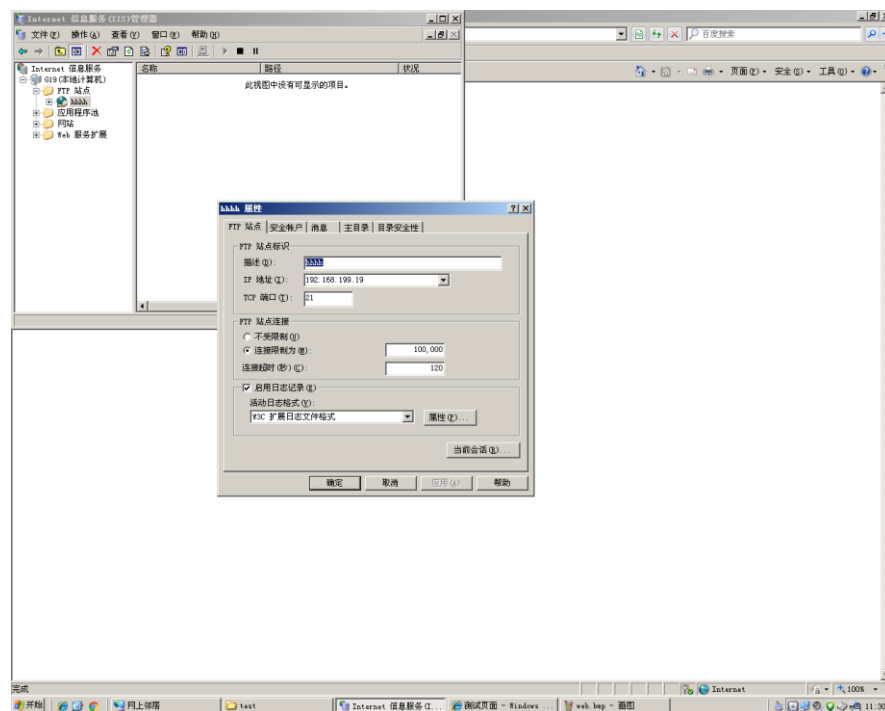
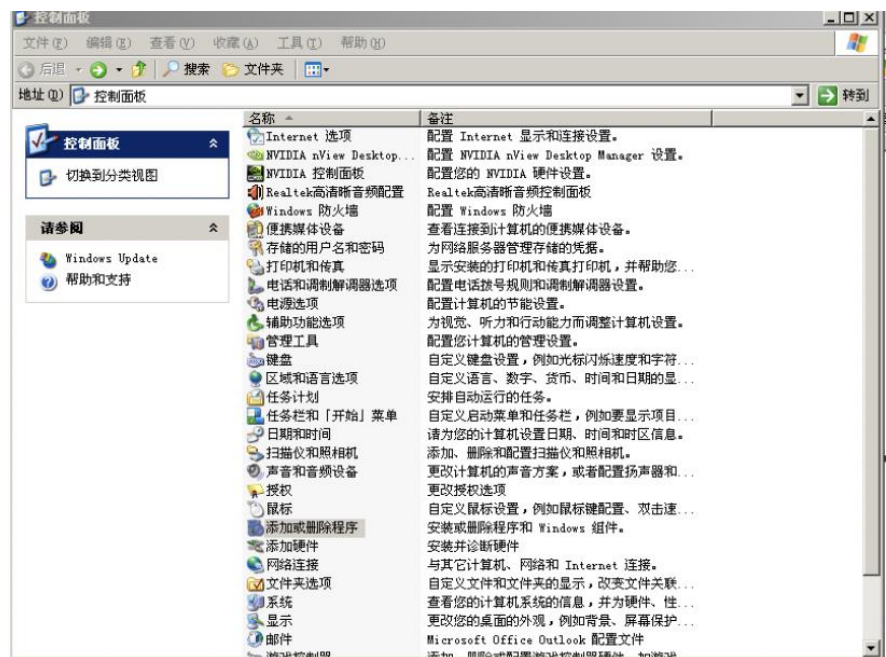


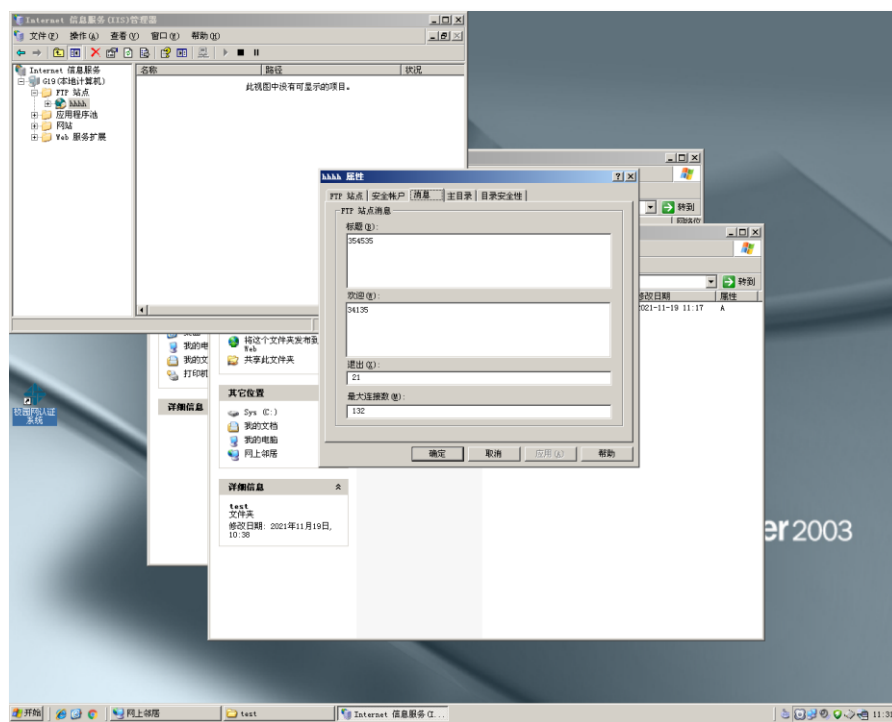
之后正常的创建区域与主机，按流程走就可以完成，此处需要创建两个主机，www 主机和 ftp 主机，到这里为止，DNS 以及 FTP 服务器的配置就完成了，接下来进行 web 端的配置选“开始”→程序→管理工具→Internet 信息服务。“默认 Web 站点”项：选“默认 Web 站点”→右键→“属性”→设置“Web 站点”：“描述”改为“测试”，“IP 地址”选择“192.168.199.18（我们做实验时的 DNS 服务器的 IP）”；“TCP 端口”维持原来的“80”不变。之后设置“主目录”：在“本地路径”通过“浏览”按钮来选择网页文件所在的目录，例如“C:\test”。设置如下：



FTP 设置步骤如下：打开“默认 FTP 站点”属性窗口：选“默认 FTP 站点”→右键→

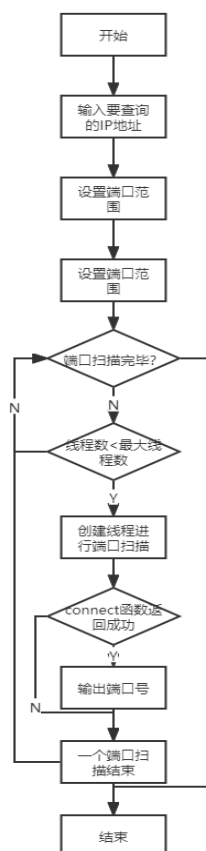
“属性”。设置“FTP 站点”：在“IP 地址”处选“192.168.199.19”，端口号保持默认值“21”不变。设置的过程如下图：





实验内容二：

流程图如下：



五、程序代码

单线程：

```
#include <stdio.h>

#include<winsock.h>

#include<string.h>

#include<iostream>

using namespace std;

#pragma comment(lib,"wsock32.lib")

int main(int argc, char* argv[])
{
    WSADATA wsaData;

    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
    {
        cout << "WSAStartup 无法初始化！" << endl;
        return 0;
    }

    // 填写远程地址信息

    sockaddr_in addr;

    addr.sin_family = AF_INET;

    while (true)
    {
```

```
char aim_ip[100];

cout << "请输入目的主机 IP 地址: ";

cin >> aim_ip;

cout << endl;

//设置获取的用户输入 IP 地址为远程 IP 地址

addr.sin_addr.S_un.S_addr = inet_addr(aim_ip);

int port_start;

int port_end;

cout << "请输入起始端口号: ";

cin >> port_start;

cout << endl;

cout << "请输入结束端口号: ";

cin >> port_end;

cout << endl;

for (int i = port_start; i <= port_end; i++)
{
    //每扫描一个端口创建一个新的套接字

    SOCKET s = ::socket(AF_INET, SOCK_STREAM,
IPPROTO_TCP);

    if (s == INVALID_SOCKET)
    {
```



```
        printf("Failed socket() %d\n", ::WSAGetLastError());

        //return -1;

    }

    //设置远程地址信息中的端口号为需要扫描的当前端口号
    addr.sin_port = htons(i);

    int ret = connect(s, (LPSOCKADDR)&addr,
sizeof(addr));

    if (ret == 0)
    {
        printf("%s:%d 端口开启\n", aim_ip, i);
    }
    else
    {
        printf("%s:%d 端口关闭\n", aim_ip, i);
    }

    ::closesocket(s);
}

cout << endl;
}

if (WSACleanup() == SOCKET_ERROR)
    cout << "WSACleanup 出错! " << endl;
```

```
    return 0;
}
```

多线程:

```
#include <winsock2.h>
#include <stdio.h>
#include <Windows.h>
#include <ws2tcpip.h>
#include <time.h>
#pragma comment(lib, "WS2_32.lib")
#pragma warning(disable:4996)
DWORD WINAPI ScanThread(LPVOID port);

int main(int argc, char* argv[])
{
    WSADATA wsd;
    int port = 0;
    int MAX_PORT;
    clock_t start, end;
    HANDLE handle;
    DWORD dwThreadId;

    //Initialize socket lib

    if (WSAStartup(MAKEWORD(2, 2), &wsd) != 0)
```

```
{

    printf("WSAStartup failed!\n");

    return 1;

}

printf("请输入要扫描的最大端口: ");

scanf("%d", &MAX_PORT);

printf("Scanning.....\n");

start = clock();

//扫描的主要代码，根据需要删减

do {

    handle = CreateThread(NULL, 0,

(LPTHREAD_START_ROUTINE)ScanThread, (LPVOID)port, 0,

&dwThreadId);

    port++;

} while (port < MAX_PORT);

WaitForSingleObject(handle, INFINITE); //等待最后一个线程结束

end = clock();

int duration = end - start;

printf("总耗时 %d ms", duration);

system("pause");

return 0;

}
```

```
DWORD WINAPI ScanThread(LPVOID port)
{
    int Port = (int)(LPVOID)port;

    int retval;//调用各种 socket 函数的返回值

    SOCKET sHost;

    SOCKADDR_IN servAddr;

    sHost = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

    if (INVALID_SOCKET == sHost)
    {
        printf("socket failed!\n");

        WSACleanup();

        return -1;
    }

    servAddr.sin_family = AF_INET;

    servAddr.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");

    //setsockopt (sHost, IPPROTO_TCP, TCP_MAXRT, (char
*)&Timeout, sizeof (Timeout)); //设置快速扫描

    servAddr.sin_port = htons(Port);

    retval = connect(sHost, (LPSOCKADDR)&servAddr,
sizeof(servAddr)); //lpsockaddr is 环路地址

    if (retval == SOCKET_ERROR) {
```

```
printf("端口%d 关闭! \n", Port); //这里不要使用
WSACleanup () 函数, 不然后续的线程会创建不了 socket

closesocket(sHost);

return -1;

}

printf("端口%d 开放! \n", Port);

closesocket(sHost);

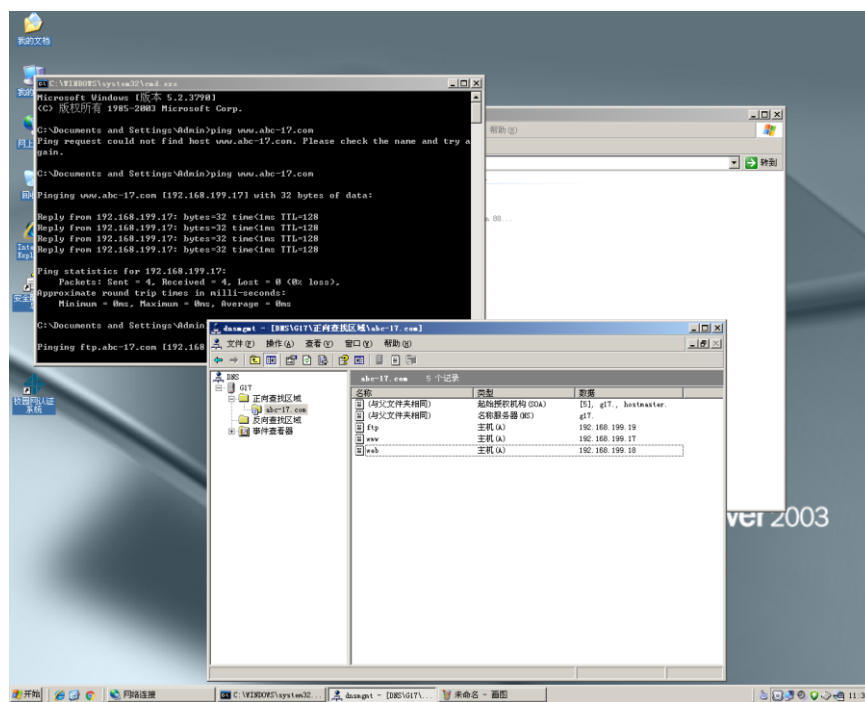
return 1;

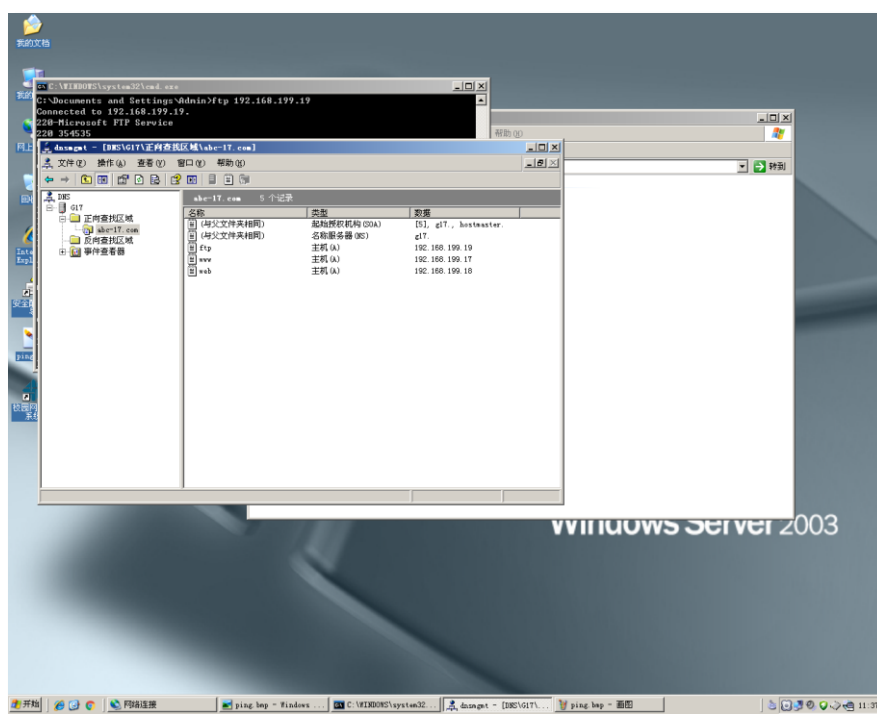
}
```

六、运行结果与分析

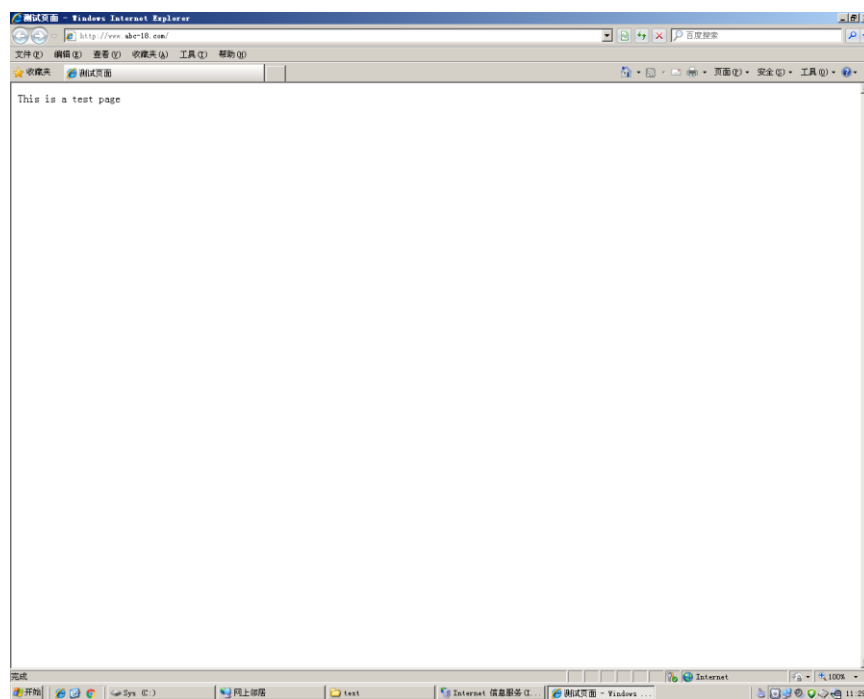
实验内容一：

1. 首先在主机上互相 ping 测试是否能连通, 从结果来看可以:

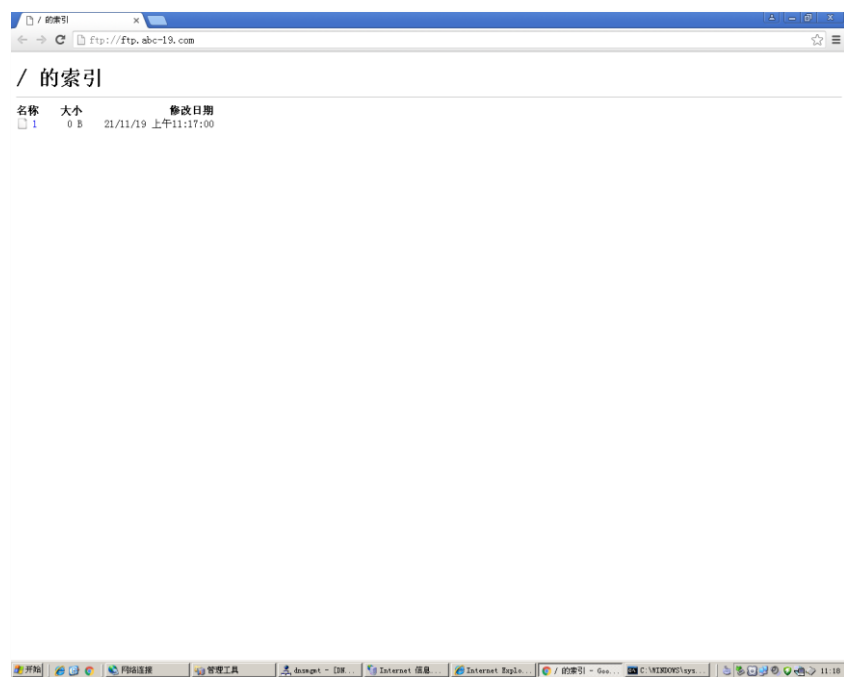




2. 在客户端的浏览器输入 www.abc-18.com, 出现了 Web 页面, 说明 Web 服务器配置成功:

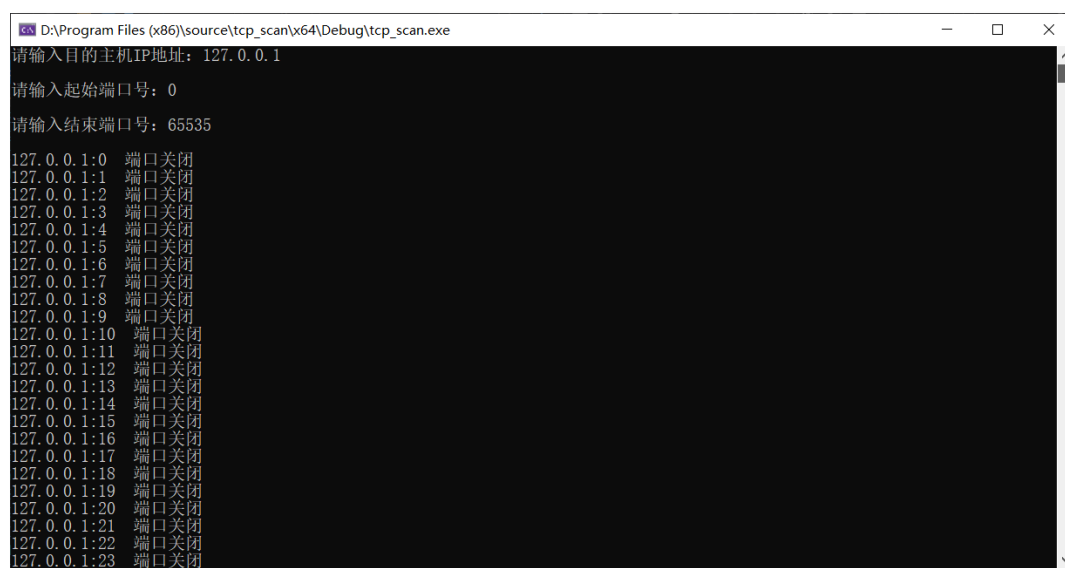


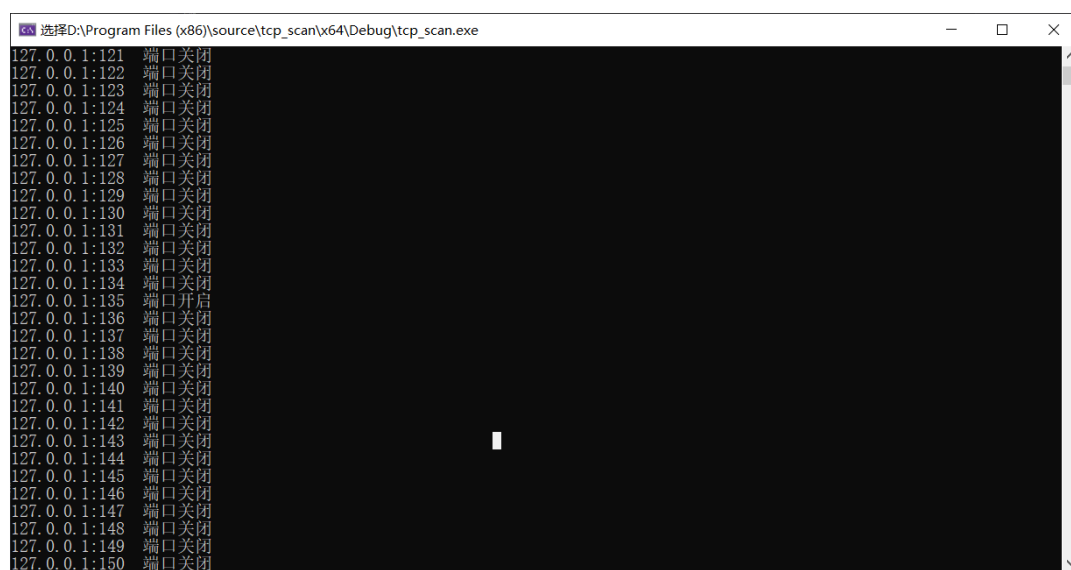
3. 在客户端浏览器输入 <ftp://ftp.abc-19.com>, 能够看到目标目录下的文件, 说明 FTP 服务器和 DNS 服务器配置成功:



实验内容二：

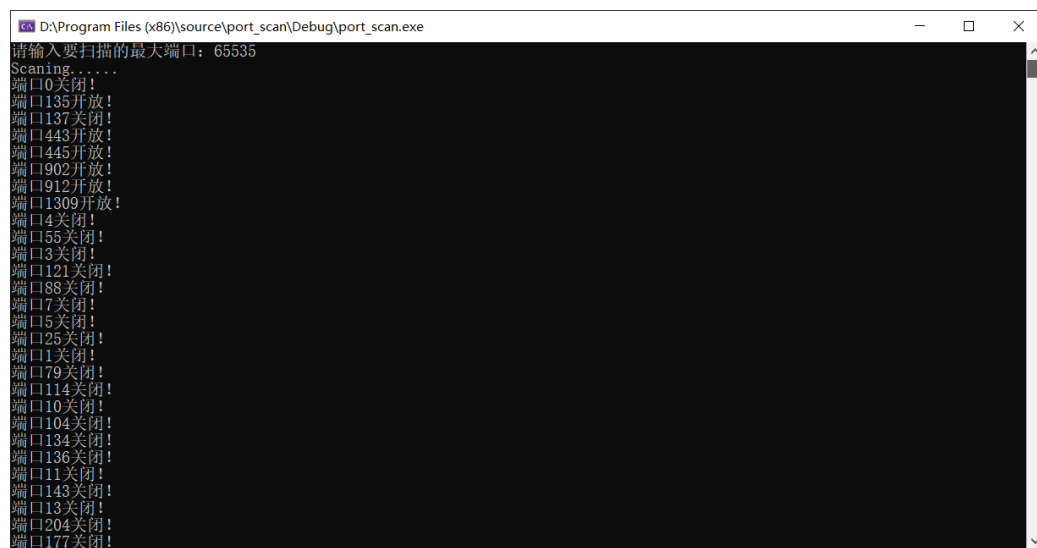
单线程 TCP 端口扫描：



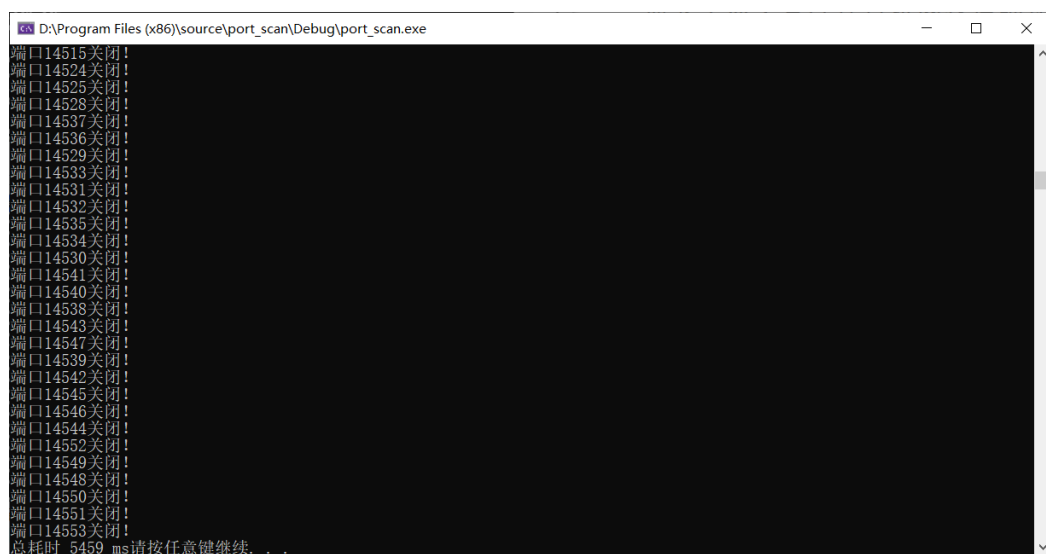


```
选择D:\Program Files (x86)\source\tcp_scan\x64\Debug\tcp_scan.exe
127.0.0.1:121 端口关闭
127.0.0.1:122 端口关闭
127.0.0.1:123 端口关闭
127.0.0.1:124 端口关闭
127.0.0.1:125 端口关闭
127.0.0.1:126 端口关闭
127.0.0.1:127 端口关闭
127.0.0.1:128 端口关闭
127.0.0.1:129 端口关闭
127.0.0.1:130 端口关闭
127.0.0.1:131 端口关闭
127.0.0.1:132 端口关闭
127.0.0.1:133 端口关闭
127.0.0.1:134 端口关闭
127.0.0.1:135 端口开启
127.0.0.1:136 端口关闭
127.0.0.1:137 端口关闭
127.0.0.1:138 端口关闭
127.0.0.1:139 端口关闭
127.0.0.1:140 端口关闭
127.0.0.1:141 端口关闭
127.0.0.1:142 端口关闭
127.0.0.1:143 端口关闭
127.0.0.1:144 端口关闭
127.0.0.1:145 端口关闭
127.0.0.1:146 端口关闭
127.0.0.1:147 端口关闭
127.0.0.1:148 端口关闭
127.0.0.1:149 端口关闭
127.0.0.1:150 端口关闭
```

多线程 TCP 端口扫描：



```
D:\Program Files (x86)\source\port_scan\Debug\port_scan.exe
请输入要扫描的最大端口：65535
Scanning.....
端口10关闭!
端口135开放!
端口137关闭!
端口443开放!
端口445开放!
端口902开放!
端口912开放!
端口1309开放!
端口4关闭!
端口55关闭!
端口3关闭!
端口121关闭!
端口88关闭!
端口7关闭!
端口5关闭!
端口25关闭!
端口1关闭!
端口79关闭!
端口114关闭!
端口110关闭!
端口104关闭!
端口134关闭!
端口136关闭!
端口11关闭!
端口143关闭!
端口13关闭!
端口204关闭!
端口177关闭!
```

综合分析可以发现单线程下的检测端口号会非常的慢,平均一个端口需要扫描 1s 左右,在老师的要求下需要扫描 1—65535 个端口大概需要很久,因此在扫描到第一个开启端口就没有再运行下去了。

同时对比可以看到,多线程端口扫描速度非常快,仅用 4 到 5 秒便可以扫描 65535 个端口。使用多线程,端口的扫描会非常的迅速,只需要几秒就可以完成。多线程下扫描的速度快于的单线程扫描的很多倍。

七、参考文献

《计算机网络原理实验指导书》 张胜兵、吕养天

《计算机网络原理实验分析与实践》 姚烨、朱怡安、张黎翔