Internet 开发技术技术

内部讲义

（2017版）

|  |  |
| --- | --- |
| 编写： | 张 绍 阳 |
| 专业： | 交通信息工程 |
| 长安大学信息工程学院 | |

目录

[序1：关于学习方式的探讨 5](#_Toc475895827)

[1. 绪论 8](#_Toc475895828)

[1.1. 软件开发知识体系梳理 8](#_Toc475895829)

[1.2. 软件设计目标 10](#_Toc475895830)

[1.3. 主流应用软件模式的演变 10](#_Toc475895831)

[1.4. 计算机软件体系结构 12](#_Toc475895832)

[1.5. 本课程的内容 15](#_Toc475895833)

[1.6. 参考资料 15](#_Toc475895834)

[1.7. 考试方式 15](#_Toc475895835)

[2. Internet编程基础---socket 16](#_Toc475895836)

[2.1. 网络进程的地址 16](#_Toc475895837)

[2.2. Socket基础知识 18](#_Toc475895838)

[2.2.1. socket基本概念 18](#_Toc475895839)

[2.2.2. 套接字的通信域 20](#_Toc475895840)

[2.2.3. Internet通信域套接字的描述 20](#_Toc475895841)

[2.2.4. Internet Socket的协议类型 20](#_Toc475895842)

[2.2.5. 三种表示Socket协议地址的结构及其用法 22](#_Toc475895843)

[2.2.6. Socket声明 23](#_Toc475895844)

[2.3. Internet通信域中Socket实现的基础技术 23](#_Toc475895845)

[2.3.1. 字节顺序 23](#_Toc475895846)

[2.3.2. 点分十进制的IP地址的转换 23](#_Toc475895847)

[2.3.3. 域名服务 24](#_Toc475895848)

[3. C/S架构软件编程技术 25](#_Toc475895849)

[3.1. 面向连接的套接字编程 26](#_Toc475895850)

[3.1.1. 工作流程 26](#_Toc475895851)

[3.1.2. 关键函数讲解 26](#_Toc475895852)

[3.1.3. 面向连接的SOCKET示例 30](#_Toc475895853)

[3.2. 关于阻塞 33](#_Toc475895854)

[3.2.1. 阻塞的基本概念 33](#_Toc475895855)

[3.2.2. 引发阻塞的函数 34](#_Toc475895856)

[3.2.3. 多线程程序 34](#_Toc475895857)

[3.3. 面向无连接的套接字编程 35](#_Toc475895858)

[3.3.1. 工作流程 35](#_Toc475895859)

[3.3.2. 面向无连接的SOCKET示例 38](#_Toc475895860)

[3.4. RAW socket套接字编程 39](#_Toc475895861)

[3.5. 上机实验 39](#_Toc475895862)

[4. B/S程序设计概述 41](#_Toc475895863)

[4.1. B/S架构应用程序工作原理 41](#_Toc475895864)

[4.1.1. 基本概念 41](#_Toc475895865)

[4.1.2. B/S架构程序的内部机理 42](#_Toc475895866)

[4.1.3. Internet程序通信方法小结 44](#_Toc475895867)

[4.2. B/S架构应用程序开发的基础 44](#_Toc475895868)

[4.2.1. Web服务器简介及架设 44](#_Toc475895869)

[4.2.2. HTTP协议 50](#_Toc475895870)

[4.2.3. HTML简介 52](#_Toc475895871)

[5. B/S应用程序开发 54](#_Toc475895872)

[5.1. 静态网页编程 54](#_Toc475895873)

[5.1.1. 基本知识 54](#_Toc475895874)

[5.1.2. 静态网页开发工具 54](#_Toc475895875)

[5.1.3. HTML标签 54](#_Toc475895876)

[5.1.4. 客户端脚本编程 56](#_Toc475895877)

[5.2. ASP动态网页编程 59](#_Toc475895878)

[5.2.1. 概述 59](#_Toc475895879)

[5.2.2. Asp关键概念 60](#_Toc475895880)

[5.2.3. 简单ASP例子 61](#_Toc475895881)

[5.2.4. ASP执行过程深入理解 61](#_Toc475895882)

[5.2.5. ASP编程基本规则 61](#_Toc475895883)

[5.2.6. ASP对象详解 61](#_Toc475895884)

[5.2.7. 完整的ASP应用程序 64](#_Toc475895885)

[5.2.8. ASP访问数据库（ADO） 65](#_Toc475895886)

[5.3. B/S程序开发流程 65](#_Toc475895887)

[5.4. 基于CSS的页面布局 66](#_Toc475895888)

[5.4.1. 布局概述 66](#_Toc475895889)

[5.4.2. CSS框模型 (Box Model) 68](#_Toc475895890)

[5.4.3. CSS 布局机制和实现方式 71](#_Toc475895891)

[6. XML语言及Ajax 79](#_Toc475895892)

[6.1. XML基础 79](#_Toc475895893)

[6.2. Ajax开发技术 79](#_Toc475895894)

[7. JSP Web开发基本架构 80](#_Toc475895895)

[7.1. Java静态页面技术支持 80](#_Toc475895896)

[7.1.1. 创建Applet 80](#_Toc475895897)

[7.1.2. 使用Applet 81](#_Toc475895898)

[7.2. JSP程序基本结构 81](#_Toc475895899)

[7.3. JSP指令和对象 83](#_Toc475895900)

[7.3.1. 编译指令 83](#_Toc475895901)

[7.3.2. 运行时指令 84](#_Toc475895902)

[7.3.3. JSP内置组件 85](#_Toc475895903)

[7.4. JSP与Java Beans 85](#_Toc475895904)

[7.4.1. Java Bean的引用 86](#_Toc475895905)

[7.4.2. Java Beans实验 87](#_Toc475895906)

[7.5. Java Servlet 89](#_Toc475895907)

[7.6. 框架简介 92](#_Toc475895908)

[7.7. 延伸学习 92](#_Toc475895909)

[8. B/S开发其他技术简介 93](#_Toc475895910)

[8.1. .Net技术 93](#_Toc475895911)

[8.1.1. .net简介 93](#_Toc475895912)

[8.1.2. .NET ASP 94](#_Toc475895913)

[8.2. SOA 及 Web Service 技术 94](#_Toc475895914)

[9. Android程序开发 95](#_Toc475895915)

[10. 后记 96](#_Toc475895916)

序：关于学习方式的探讨

人类进入信息时代，知识的记录、存储和检索方式发生了重大的改变。对大学教育也提出了相应的挑战。人才培养的本质不再是知识传递，而应该是知识运用能力和学习知识的能力。

1. 以知识传递为主线的教育已不能适应信息时代教学的需要。

印刷术发明以前，知识主要靠手抄、口授单对单进行传递，印刷术的发明，使知识传递速度大大加快，但是其检索仍然是困难的。学习知识的人需要到图书馆等知识集中的地方进行查阅，学习。每个图书馆不可能将所有图书进行收藏，为了查找自己所需的知识，研究者需要花费大量的时间精力进行图书检索，其效率是非常低下的。这个阶段，我们赞扬一个人往往说他“学识渊博”，是说他知识很丰富，懂得东西多。这个阶段，求学的主要目的是获取知识。传统上，我们将求学的人叫做“学生”。

计算机及互联网络普及，改变了知识的获取方式。我们轻敲键盘，全人类几千年文明所积累的知识都可以展现在我们眼前。多种智能的搜索方式，使得我们能够尽快得到我们所需要的知识。但是，目前大多数大学教学本质上仍延续古代私塾的方式，传道、授业、解惑，其核心是进行知识的传递。只不过形式上进行了一些改进，从教师写黑板到现在用PPT的方式讲解。相信很多老师都有同感，上课学生听课的少，老师也在疑惑，现在的学生怎么了，什么都不懂，还不好好学，他们到底想干什么？学生也在抱怨，老师讲的东西书上、网上或者精品课程资源中说的很明白，学了有什么用？我认为，其核心问题是教师的知识获取和学生的知识获取渠道相同，知识已经不是稀缺资源，所以学生对知识的渴求没有。教师在课堂上所讲授的专业基本知识到处都是，教师所传达的信息量对学生没有吸引力。图1显示了传统知识获取和现代知识获取的渠道及其中教学双方的关系。

当然，我们相信教师在专业领域具有深入的研究和广博的知识，但是在课堂上的讲授仍然只能是基本知识，有经验或知识面广的教师讲的更透彻，但是，往往是具有对等水平的听众才能深入体会，作为初学者，其所获得的信息量是差不多的。再者，教师的水平再高，其逻辑性也不一定比书的作者反复修改、认真雕琢的逻辑性强。

所以说，以知识传递为主线的教育教学模式已不能适应信息时代学习的需求。

学习

学习

知识运用能力

知识学习能力

学习

知识

知识

……

老师

学生

传递知识

知识库

老师

学生

信息时代学习模式

传统学习模式

图1 学习模式的改变

1. 信息时代社会对知识的需求发生了改变

随着信息学科和工业化的飞速发展，局部的简单的机械性、重复性的工作逐步被自动化工具所取代，计算机软件帮助人们提高效率。例如,在信息学科,高集成芯片的发展使得硬件设计越来越简单和流程化，软件开发语言和工具的发展使得普通信息系统的开发也称为组件式的组合。IT企业成为一种低门槛的企业，高中生经过几个月的培训即可上岗工作，基本能力是技能，属于职业技能，不是大学教育的主要目的。IT企业在解决效率方面的工作已经不是盈利的主流，因为效率大家都能看得见，因此，竞争就比较激烈。企业要生存，企业之间的竞争不是公司所积累的知识的竞争，而称为一种创新性的竞争。只有创新，才能够形成别人所不具备的能力。

因此，企业真正缺少的往往不是执行人员，而是具有创新性思维的

什么是创新性？我认为：创新性就是提出问题，运用知识解决问题的能力。

1. 终身学习方式要求教学应以学习能力培养为主

每个学科都在快速发展，人类的知识在不断的更新，注定在每个行业，我们都要不断的学习，没有人在大学中就将知识学完了，在职业道路上，大学所学只是沧海一粟了。因此，教学应以学科特点，培养学生自学能力，使其能够快速把握学科方向，掌握学习方法。

综合上述几个问题，将信息时代的知识学习不再是以知识传递为主要目的，而是一种**以创新能力培养为目标的知识运用能力的培养**。在运用知识解决问题的过程中，潜移默化传递一种学习能力。

基于新的学习模式，大学教育应发生重大改革，包括以下几个方面：

1. 学习方式的变化

传统学习模式在学校是学习，走到社会工作后运用所学知识解决问题。在新的学习模式中，知识运用能力的培养是主流，因此，学习变为以问题为引导，在解决问题中学习，能够充分发挥学习的主动性。

1. 教材的变化

教材不再是知识的堆积，而应该是一种面向问题的解决方法的引导；教材提出成成中的典型问题以及学生能达到的目标，给出一种学习基础知识的引导和解决思路的引导。

1. 教师的变化

教师是问题的提出者、思路的引导者、新的问题的开拓者。

1. 学生的变化

学生是学习的主体，以被动学习改变为目标驱动的主动学习；

1. 课堂的变化

课堂不再是老师讲解的模式，而是一种在协作基础上的讨论式的交流。

本教材正是在以上的总体指导思想下编写的，不足之处，请大家多多指正。本教材适用于高等学校计算机科学与技术专业或者软件专业学生学习Internet程序开发，学生自学或者在教师指导下的以自我为主的学习方式。

1. 绪论
   1. 软件开发知识体系梳理

本门课讲的是Internet编程技术，属于编程技术的范畴，首先对学过的编程技术做一个简要回顾，在此基础上讲解本门课开设的意义、方法、内容。

在大多数学校中计算机科学与技术专业的学习，都是从《程序设计基础》开始，其中结构化的程序设计方法一般都是以C语言为基础进行讲解。有的专业学生基础比较好，直接从C++开始，但一般也会普及一下C语言的基础。在C语言中，我们学会使用结构化程序的基本编程方法。接下来，在《算法与数据结构》中，应熟练应用C语言进行算法实现，从而打好结构化编程的基础。

在《面向对象的程序设计》中，学习了面向对象的基本思想，会编写图形用户界面，会抽取类并运用面向对象的方法进行基本的程序设计。在《Java程序设计》中会进一步深入学习面向对象的思想，学习软件分层的基本概念，学会如何去组织类。通过这两门课的学习，进入到面向对象的编程世界。

在《数据库原理与应用》课程中，掌握数据库的基本原理和应用方法，掌握数据建模，能够使用开发工具访问数据库。

通过以上课程的学习，初步学会了控制台应用程序、桌面应用程序、数据库应用程序的编写。

学到此处，大家具有了一定的编程能力，请回答下面的几个问题：



1. 软件设计的目标是什么？根据学习语言的经验试着回答。
2. 软件还有那些种类？请大家举例，见过的软件类型。
3. 软件编程技术和语言之间的关系？



* 1. 软件设计目标

正确性：能够满足需求的开发，称为是正确的。

健壮性：具有容忍错误能力的应用程序，称为是健壮的。

灵活性：能够增加同类型或者不同类型的功能，以及改变功能。

可重用性：代码、方法、模块等是可重用的。

高效性：时间效率：在指定的时间内完成；空间效率：内存或外存的限制。

提示：设计目标应牢记在心。



* 1. 主流应用软件模式的演变

1. 主机/哑终端模式集中计算模式

“主机／终端”模式是早期采用的工作模式，用户计算机远程登录服务器主机后，用户计算机以仿真终端的形式工作；用户通过键盘发出命令请求该命令被送到服务器，服务器主机执行命令，并将命令结果返回用户计算机的显示器上显示。在这种模式下，用户所用终端为标准输入设备（键盘）与标准输出设备（显示器），本身不执行命令，全部命令在远程服务器主机上执行。时至今日，因特网上某些公共信息服务，例如telnet服务、BBS等还保留着这种服务模式。企业应用包括超市收银，航空售票，小型储蓄所等。上世纪80年代出现的一些小型机、中型机都是这种模式。

1. 单机模式

在单台计算机上进行计算、信息管理的软件模式。随着上世纪90年代初X86系列PC机的兴起，硬件集成度的快速提升，个人计算机应用软件进入一个繁荣的时代。个人的应用需求快速得到满足。

1. 客户机/服务器软件模式

20世纪90年代以来，由于计算的需要、协同工作的需要，客户机/服务器模式（Client/Server，C/S）已成为越来越广泛使用的一种新型计算机应用模式。其基本结构是指把一个大型的计算机应用系统变为多个能互为独立的子系统，服务器是整个应用系统资源的存储与管理中心，多台客户机则各自处理相应的功能，共同实现完整的应用。

Client程序的任务是将用户的要求提交给Server程序，再将Server程序返回的结果以特定的形式显示给用户；Server程序的任务是接收客户程序提出的服务请求，进行相应的处理，再将结果返回给客户程序。

C/S结构的优点：可以充分发挥客户、服务器两端硬件环境的优势，将任务合理分配到 Client端和Server端来实现，提高响应和处理速度，降低了系统的通讯开销。

缺点主要有以下几个：客户端需要安装专用的客户端软件。首先涉及到安装的工作量，其次任何一台电脑出问题，如病毒、硬件损坏，都需要进行安装或维护。还有，系统软件升级时，每一台客户机需要重新安装，其维护和升级成本非常高。

1. 浏览器/服务器应用模式

B/S结构，即Browser/Server(浏览器/服务器)结构，就是只安装维护一个服务器(Server)，而客户端采用浏览器(Browse)运行软件。

在B/S体系结构系统中，用户通过浏览器向分布在网络上的许多服务器发出请求，服务器对浏览器的请求进行处理，将用户所需信息返回到浏览器。而其余如数据请求、加工、结果返回以及动态网页生成、对数据库的访问和应用程序的执行等工作全部由Web Server完成。

B/S结构的主要特点是维护方便、开发简单且共享性强、总体拥有成本低。但数据安全性问题、对服务器要求过高、数据传输速度慢、软件的个性化特点明显降低，这些缺点是有目共睹的。尤其是难以实现实时系统的要求。

1. 云计算模式

在2006年，Google and IBM宣布联合开发一种技术，其目标定位为：“the emerging paradigm of large-scale distributed computing,” also known cloud computing。

一种广泛接受的定义：云计算是通过网络提供可伸缩的廉价的分布式计算能力。用户可以通过已有的网络将所需要的庞大的计算处理程序自动分拆成无数个较小的子程序，再交由多部服务器所组成的更庞大的系统，经搜寻、计算、分析之后将处理的结果回传给用户。比如搜寻引擎、网络信箱等。

广义的云计算是指服务的交付和使用模式，指通过网络以按需、易扩展的方式获得所需的服务。这种服务可以是IT和软件、互联网相关的，也可以是任意其他的服务。

狭义的云计算是指IT基础设施的交付和使用模式，指通过网络以按需、易扩展的方式获得所需的资源(硬件、平台、软件)。提供资源的网络被称为“云”。“云”中的资源在使用者看来是可以无限扩展的，并且可以随时获取，按需使用，随时扩展，按使用付费。这种特性经常被称为像水电一样使用IT基础设施。云服务可以将企业所需的软硬件、资料都放到网络上，在任何时间、地点，使用不同的IT设备互相连接，实现数据存取、运算等目的。

当前，常见的云服务有**公共云**(Public Cloud)与**私有云**(Private Cloud)两种。公共云还可细分为3个类别，包括Software-as-a- Service, SaaS(软件即服务)、Platform-as-a-Service, PaaS(平台即服务)及Infrastructure-as-a-Service, IaaS(基础设施即服务)。



延伸思考：是不是似曾相识？你认为应用软件会向那个方向发展？

* 1. 计算机软件体系结构

上述是主流的应用软件模式的演变过程。本小节从软件设计角度，即技术角度，对计算机软件体系结构进行分类和说明。

计算机软件体系结构的定义：可重用的、软件各部分的组织方法。

为什么要有体系结构：最大的目的是实现可重用性。

Eric J. Braude[[1]](#footnote-1)将体系结构分为如图1所示的几个部分。

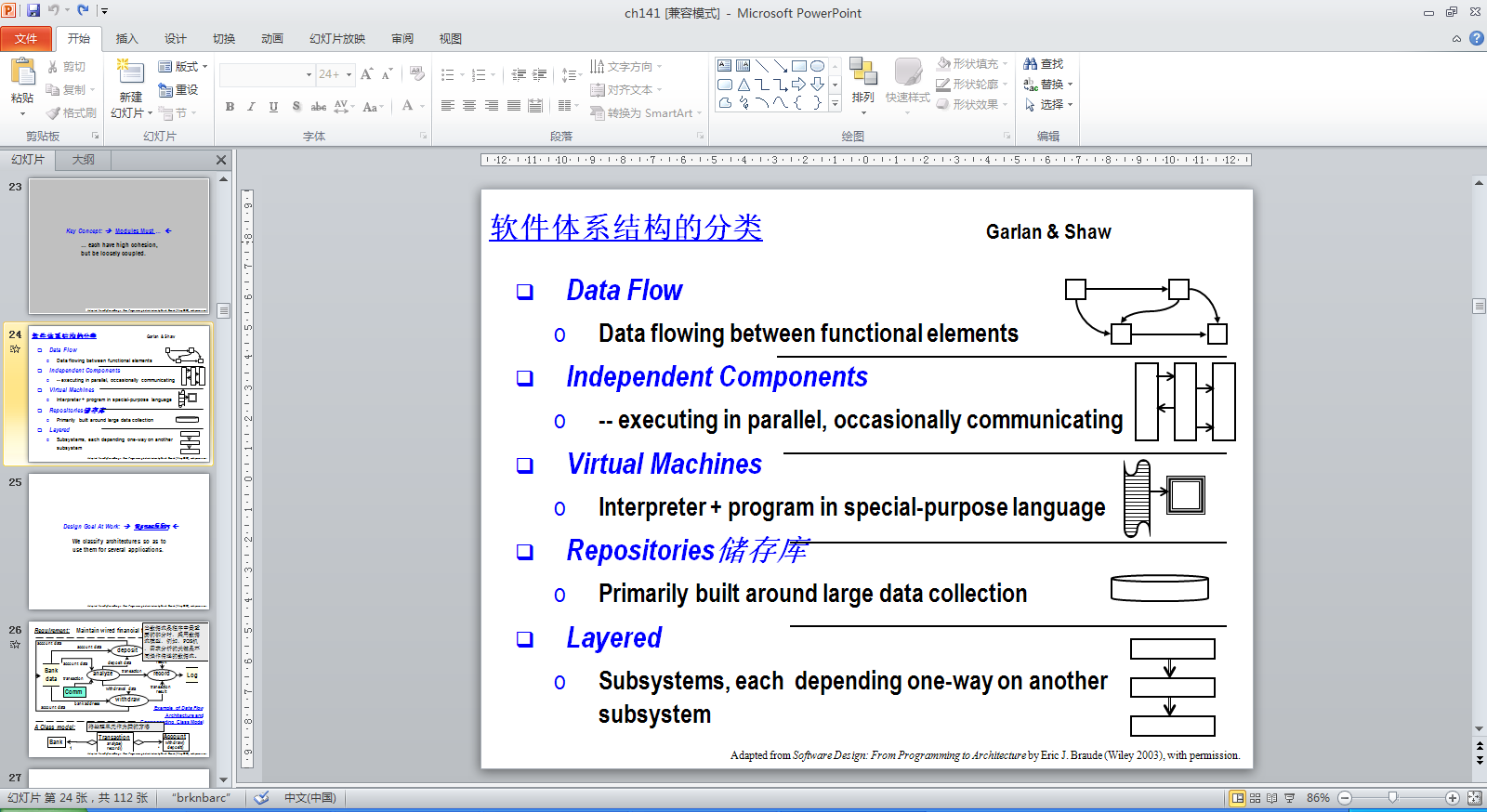


图1 软件体系结构分类

1. 数据流

当数据是程序中最重要的部分时，采用数据流模型，例如，POS机，需求分析的关键是不同功能元素之间传递的数据流。

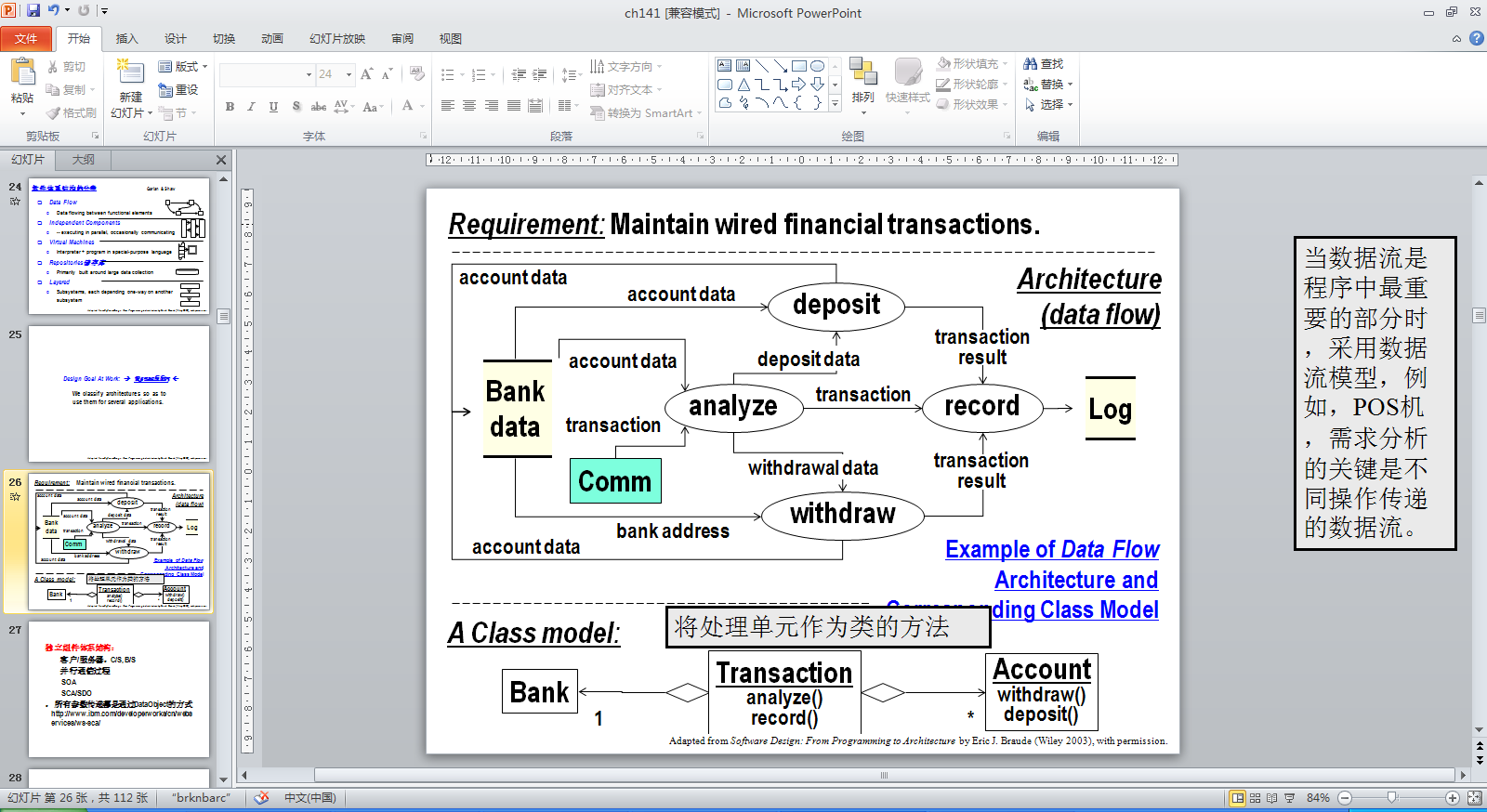


图2 POS机软件的体系结构

1. 独立组件体系结构

独立组件的关键概念是组件各自并行执行，必要时进行通信，网络应用程序、单机应用都可以是独立组件结构。例如：

* 客户/服务器软件，C/S, B/S
* 并行通信软件
* SOA，SCA/SDO，所有参数传递都是通过DataObject的方式传递。图3是IBM提出的SCA组件模型[[2]](#footnote-2)。

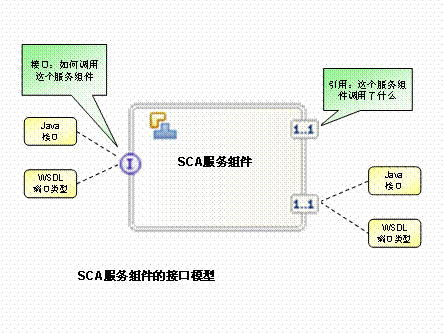


图3 SCA服务组件模型

1. 虚拟机

如果软件需要某种专门的语言来实现，需要使用虚拟机。例如：java虚拟机。Java虚拟机（JVM）是Java Virtual Machine的缩写，它是一个虚构出来的计算机，是通过在实际的计算机上仿真模拟各种计算机功能模拟来实现的。Java虚拟机有自己完善的硬件架构，如处理器、堆栈、寄存器等，还具有相应的指令系统。

Java语言的一个非常重要的特点就是与平台的无关性。而使用Java虚拟机是实现这一特点的关键。一般的高级语言如果要在不同的平台上运行，至少需要编译成不同的目标代码。而引入Java语言虚拟机后,Java语言在不同平台上运行时不需要重新编译。Java语言使用模式，Java虚拟机屏蔽了与具体平台相关的信息，使得Java语言编译程序只需生成在Java虚拟机上运行的目标代码(字节码)，就可以在多种平台上不加修改地运行。Java虚拟机在执行字节码时，把字节码解释成具体平台上的机器指令执行。

思考：有没有其他的虚拟机？



问题提示：当编程语言没有被编译成为计算机可执行的目标代码，语言需要解释执行，都需要虚拟机，.net平台的Framework也是采用了虚拟机的概念。

1. 数据库软件

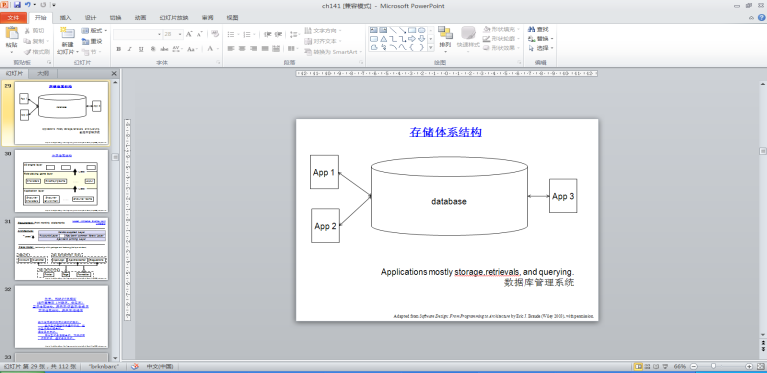


图4 数据库软件体系结构

以数据库为核心的应用程序。重点在于数据的管理、维护、展示和应用等工作内容。

1. 层次体系结构

是指软件的结构是分层次的。下面是常见的层次结构：

* OSI规定的计算机网络的7层模型；
* 传统的Visual Basic应用程序，分为代码层、类库层；
* 应用程序的三层体系结构：表示层/逻辑层/数据访问层
* 两层体系结构：表示层/数据层

独立组件体系结构和层次结构的区别：

* 组件之间通过信息进行交流，组件之间是松耦合的。通常是双向的。
* 层次之间是紧耦合的，方法调用，依赖较多，通常是单向的。

在一个软件系统中，多种体系结构都可能同时出现，例如独立组件结构的软件系统，其每个组件可以是分层次的；数据库应用程序也可以是分层次的。说某个软件系统是某种结构，是从其大的结构或者观察者感兴趣的角度来说的，各种结构的运用都是为了保证软件的设计目标的实现。

* 1. 本课程的内容

本门课重点介绍Internet程序的开发，属于软件开发技术。应该归类到独立组件体系结构的软件系统。独立组件的特点就是独立运行，必要时通信。因此，讲解的重点是Internet程序组件之间的的通信。

本门课的内容包括：

* 互联网编程基础，socket
* C/S程序开发，讲解基于客户端和服务器端应用程序的编程；
* B/S程序开发，讲解B/S程序的原理、静态网页编程技术、动态网页编程、编程新技术，包括XML、框架、Ajax开发技术、SOA 及 Web Service 技术等；
* 移动互联编程技术。

目的：通过本门课程的学习，对软件体系结构有清晰的认识；深入理解Internet编程的基础；培养快速进行网络编程技术的学习的能力；具备Internet应用程序开发的基本能力。

* 1. 参考资料

<http://www.w3school.com.cn/>

<ftp://202.117.67.94>

* 1. 考试方式

试卷（开卷）：60%，简答题和编程题为主；

上机实验：30%，按要求自我调试完成并经过验收；

到课率：10%。

1. Internet编程基础---socket

Internet 程序的最大特征就是能够在互联网上相互通信，完成复杂的协同任务。要通信，就应该有地址。

* 1. 网络进程的地址

复习1：计算机中的地址

常见计算机中的地址包括三类：

1. Mac地址：属于Ethernet网络的数据链路层地址。用于在以太网中进行寻址。为什么要有Mac地址？
2. IP地址：TCP/IP网络中的计算机地址。为什么要有IP地址？回顾一下网络层的作用。

有了IP地址就可以通信吗？答案是否定的。仅有IP地址是不够的，现在的计算机都是多任务的操作系统，我们需要知道每个“程序”的地址。所以需要第三个地址，即网络进程地址。



1. 网络进程的地址:用于标识网络中的进程的方法。

复习2：进程

*进程的概念*：是一个具有一定独立功能的程序关于某个数据集合的一次运行活动。它是操作系统动态执行的基本单元，在传统的操作系统中，进程既是基本的分配单元，也是基本的执行单元。进程的概念主要有两点：第一，进程是一个实体。每一个进程都有它自己的地址空间，一般情况下，包括文本区域（text region）、数据区域（data region）和堆栈（stack region）。文本区域存储处理器执行的代码；数据区域存储变量和进程执行期间使用的动态分配的内存；堆栈区域存储着活动过程调用的指令和本地变量。第二，进程是一个“执行中的程序”。程序是一个没有生命的实体，只有处理器赋予程序生命时，它才能成为一个活动的实体，我们称其为进程。

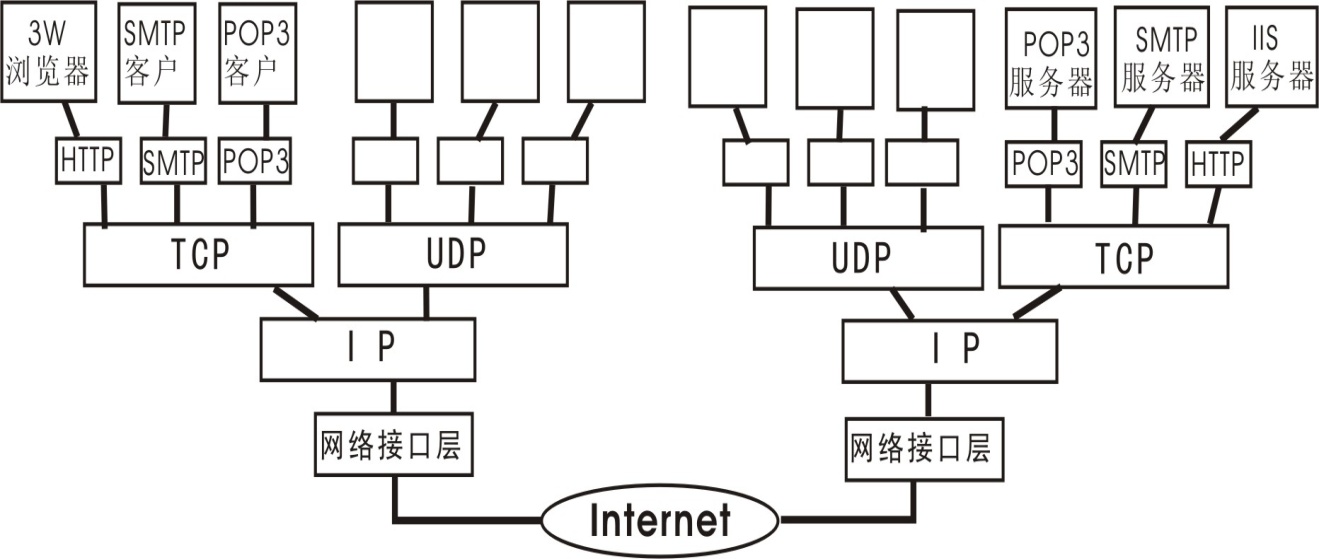
*进程和线程的关系*：通常在一个进程中可以包含若干个线程，它们可以利用进程所拥有的资源。在引入线程的操作系统中，通常都是把进程作为分配资源的基本单位，而把线程作为独立运行和独立调度的基本单位。由于线程比进程更小，基本上不拥有系统资源，故对它的调度所付出的开销就会小得多，能更高效的提高系统内多个程序间并发执行的程度。

下图是互联网中应用程序的通信模型。从中可以看出，互联网通信实际上是两个计算机中的某两个进程在通信。相对于单机操作系统中的进程概念，我们将互联网中两个通信的进程称为*网络进程*。

图2 互联网应用程序通信模型

计算机A

计算机B



进程A

进程B

那么，如何标识网络进程？

---在TCP/IP协议族中，使用传输层端口号来进行标识。

问题：什么叫端口？

*端口*是TCP/IP协议族中应用层进程与传输层协议之间的通信接口。每个端口都拥有一个的整数型的标识符，称作*端口号*（port number）。端口号类似于文件描述符。

大家还记得TCP和UDP数据包的格式。如图2-1所示。

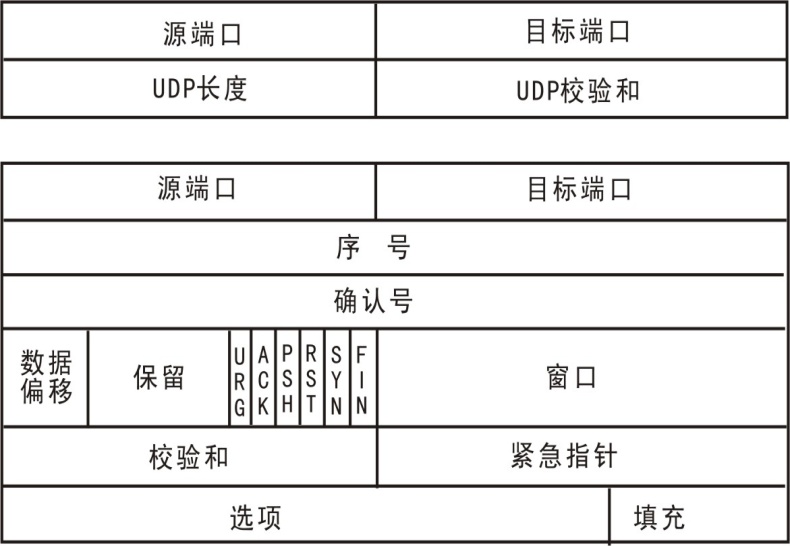


图2-1 UDP和TCP协议数据包的格式

在协议中就有端口号。指明该数据包是给那个端口的。大家还记得常用网络应用程序的端口号，FTP---20，21；telnet---23；www---80。并且，两种协议的端口号是单独编号的。

思考：这样的数据包能到达目的地吗？好像没有地址啊，为什么？

因此，网络进程的标识至少包括三个因素，IP地址指示该进程位于某台计算机，传输层协议表明该进程采用的协议类型，端口号表明其序号。

用集合的方法，可用一个三元组唯一地标识一个应用层网络进程地址：

**应用层网络进程地址=（传输层协议，主机的IP地址，传输层的端口号）**

问题：在某台计算机上安装了FTP服务、WWW服务，这些网络进程的标识是什么？如何访问？

通常我们在访问互联网进程的时候没有输入其端口号，为什么？

为什么是应用层？

延伸思考：如果从外网访问内网的某个应用程序或者服务，能否直接访问，应该如何做才行？

* 1. Socket基础知识

上面从直觉角度知道了计算机网络通信的基本需要和地址，下面我们系统了解一下网络通信的实现方法。

* + 1. socket基本概念

加州大学伯克利(Berkley)分校开发并推广了一个包括TCP/IP互联协议的UNIX，称为BSD UNIX（Berkeley Software Distribution UNIX）操作系统，在这个操作系统中，使用了一个叫做套接字的技术，用来实现进程之间的通信，包括网络进程。后来的许多操作系统没有另外搞一套网络通信方法，而是选择了对套接字的支持。由于这个套接字规范最早是由Berkeley大学开发的，一般将它称为Berkeley Sockets规范。

Socket英文即插座的意思，在计算机中也是一样，图2-2是我们常用的插座。

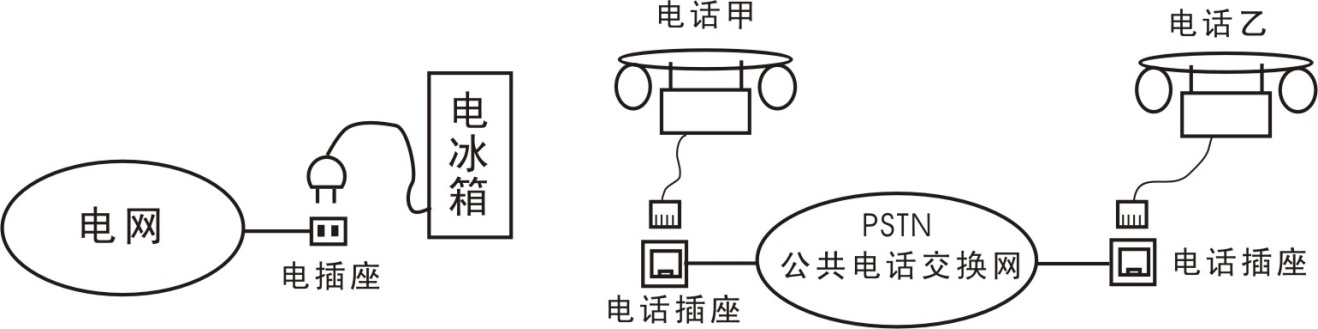


图2-2 套接字直观理解

在网络通信中，套接字上联应用进程，下联网络协议栈，是应用程序通过网络协议栈进行通信的接口，是应用程序与网络协议栈进行交互的接口。如图2-3所示。

**如下图所示：**

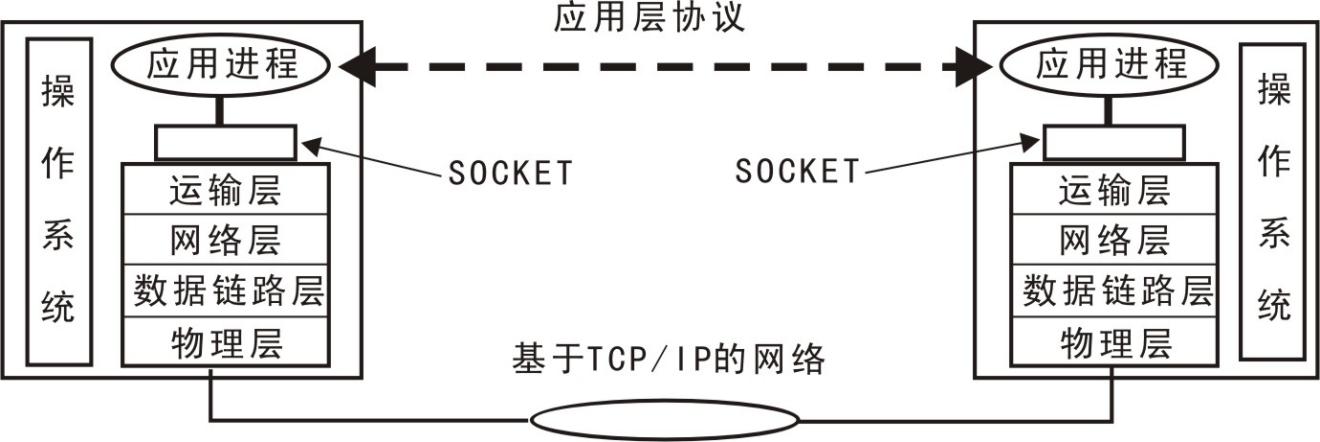


图2-3 Internet中Socket所处的位置

从实现的角度来讲，套接字是一个复杂的软件机构，包含了一定的数据结构，包含许多选项，由操作系统内核管理。

从使用的角度来讲，套接字非常简单。操作系统一般对于套接字的操作形成了一种网络应用程序的编程接口（socket API）。它定义了应用程序与协议栈软件进行交互时可以使用的一组操作，决定了应用程序使用协议栈的方式、应用程序所能实现的功能。

总之，套接字是网络通信的基石。

* + 1. 套接字的通信域

但是，历史的和现实的网络并不仅仅由Internet构成，所以，套接字在最初设计的时候考虑了网络协议的不同，提出通信域概念。

通信域即指基础的通信协议，套接字通常只和同一通信域中的套接字交换数据。例如AF\_INET、AF\_INET6、AF\_LOCAL（或称AF\_UNIX），如AF\_INET决定了在ipv4网络中进行通信。AF\_UNIX决定了要在UNIX环境中进行通信。

如果数据交换要穿越通信域的边界，就一定要执行某种解释程序。所以套接字的描述中还需要包含通信域的描述。

现在，绝大多数应用仅针对Internet域，使用Internet 协议族（即TCP/IP协议族）来通信。下面主要以Internet通信域进行讲解。

* + 1. Internet通信域套接字的描述

总结以上所讲，在Internet套接字描述中应该包含以下内容：

* 通信域；
* 套接字的协议类型（数据报、流式）；
* 套接字协议地址（IP地址，传输层端口号）

为了表示更广泛的通信域，套接字协议地址中没有包含上面所讲的三元组中的协议类型，在实现的时候分为两步，首先定义地址，在套接字句柄中定义协议类型，双方进行关联。

* + 1. Internet Socket的协议类型

Internet传输层包括TCP和UDP两种协议，对应的套接字也包含两种，数据报和流式套接字，另外，为了便于底层访问，规范中还给出了原始套接字，用于进行IP层的直接访问。

（1）数据报套接字（Datagram SOCKET）

数据报套接字提供无连接的不保证可靠的独立的数据报传输服务。在Internet通信域中，数据报套接字使用UDP数据报协议形成的进程间通路，具有UDP协议为上层所提供的服务的所有特点。

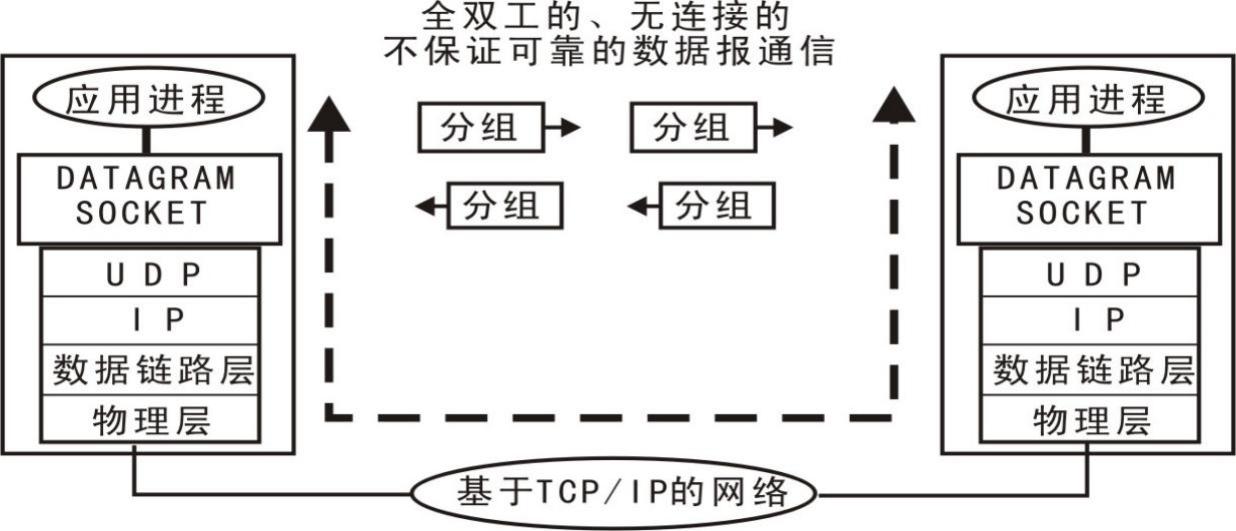


图 在Internet通信域中，数据报套接字基于UDP协议

（2）流式套接字（Stream SOCKET）

流式套接字提供双向的、有序的、无重复的、无记录边界的可靠的数据流传输服务。在Internet通信域中，流式套接字使用TCP协议形成的进程间通路，具有TCP协议为上层所提供的服务的所有特点，在使用流式套接字传输数据之前，必须在数据的发送端和接收端之间建立连接，如下图所示。

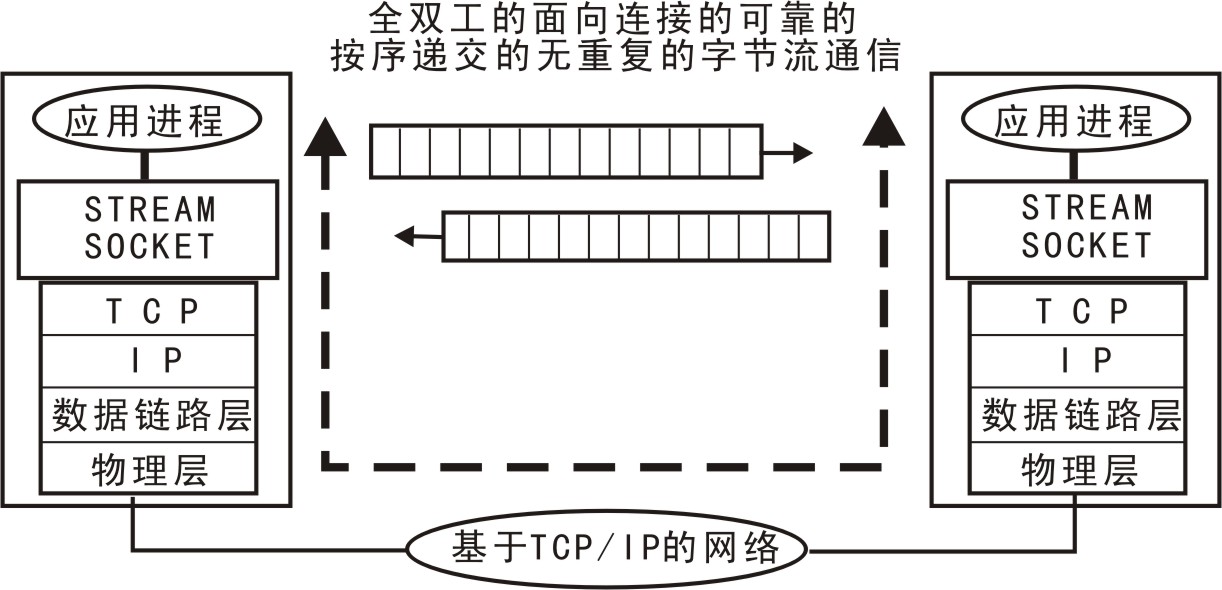


图 基于TCP的通信模型

（3）原始式套接字（RAW SOCKET）

原始式套接字允许对较低层次的协议，如IP、ICMP直接访问，用于检验新的协议的实现。原始套接字不需要传输层端口。

延伸思考：至此，我们知道了传输层和IP层的访问方法，大家可能有这疑问，数据链路层能否访问呢？如果能够编程，那么是否就可以通过链路层进行互联网通信呢？

回答提示：能够访问，window下的参考资料，WinPcap：<http://baike.baidu.com/view/696423.htm>

* + 1. 三种表示Socket协议地址的结构及其用法

在套接字编程接口中，专门定义了三种结构型的数据类型，用来存储协议相关的协议地址，在套接字编程接口的函数调用中要用到它们。

（1）sockaddr\_in结构，专门针对Internet通信域，存储套接字相关的协议地址信息，例如IP地址，传输层端口号等信息。

struct sockaddr\_in {

short int sin\_family; // 地址家族

unsigned short int sin\_port; // 端口号

struct in\_addr sin\_addr; // IP 地址

unsigned char sin\_zero[8]; // 全为0

}

（2）in\_addr结构，专门用来存储 IP地址。

Struct in\_addr {

Unsigned long s\_addrl;

}

（3）sockaddr结构，针对各种通信域的套接字，存储它们的地址信息。

struct sockaddr {

unsigned short sa\_family; // 地址家族，即通信域

char sa\_data; // 协议地址

}

（4）使用方法

首先，定义一个Sockaddr\_in的结构实例，并将它清零。

比如：struct sockaddr\_in myad;

memset(&myad,0,sizeof(struct sockaddr\_in));

然后，为这个结构赋值，

比如：myad.sin\_family=AF\_INET;

myad.sin\_port=htons(8080);

myad.sin\_addr.s\_addr=htonl(INADDR-ANY);

第三步：在函数调用中使用时，将这个结构强制转换为sockaddr类型。

如：accept(listenfd, (sockaddr\*)(&myad), &addrlen);

* + 1. Socket声明

int SOCKET( int domain, int Type, int Protocol);

例如：sockfd=SOCKET(AF\_INET, SOCK\_STREAM, 0)

在这里没有指定本地跟远程的协议地址。在应用的时候再将socket和地址进行关联。方法在其他函数中进行讲解。

* 1. Internet通信域中Socket实现的基础技术
     1. 字节顺序

在计算机中的多字节数据的存储顺序，称为本机字节顺序。

多字节数据在网络协议报头中的存储顺序，称为网络字节顺序。

网络应用程序要在不同的计算机中运行，本机字节顺序是不同的，但是，网络字节顺序是一定的。

所以，应用程序在编程的时候，在把IP地址和端口号装入套接字的时候，应当把它们从本机字节顺序转换为网络字节顺序；相反，在本机输出时，应将它们从网络字节顺序转换为本机字节顺序。

套接字编程接口特为解决这个问题设置了四个函数：

htons() 短整数本机顺序转换为网络顺序，用于端口号。

htonl() 长整数本机顺序转换为网络顺序，用于IP地址。

ntohs() 短整数网络顺序转换为本机顺序，用于端口号。

ntohl() 长整数网络顺序转化为本机顺序，用于IP地址。

这四个函数将被转换的数值作为函数的参数，函数返回值是转换后的结果。

* + 1. 点分十进制的IP地址的转换

在因特网中，IP地址常常用点分十进制的表示方法，但在套接字中，IP地址是无符号的长整型数，套接字编程接口设置了两个函数，专门用于两种形式的IP地址的转换。

（1）inet-addr函数

unsigned long inet-addr( const char\* cp)

入口参数cp：点分十进制形式的IP地址。

返回值： 网络字节顺序的IP地址，是无符号的长整数，

（2）inet\_ntoa函数

char\* inet\_ntoa(struct in\_addr in)

入口参数in：包含长整型IP地址的 in\_addr 结构变量。

返回值： 指向点分十进制IP地址的字符串的指针。

* + 1. 域名服务

通常，我们使用域名来标识站点，可以将文字型的主机域名直接转换成IP地址，

struct hostent\* gethostbyname( const char\* name);

入口参数：是站点的主机域名字符串，

返回值： 是指向hostent 结构的指针，

hostent结构包含主机名，主机别名数组，返回地址的类型（一般是AF-INET），地址长度的字节数，已符合网络字节顺序的主机网络地址等。

1. C/S架构软件编程技术

C/S架构软件即传统的客户服务器架构软件，无论客户端和服务器端本身如何工作，客户端和服务器的通信都是使用Socket实现。本节课讲客户端和服务器端通信的几种方式，即Socket的几种工作模式的实现。

UNIX操作系统对文件和所有其它的输入/输出设备采用一种统一的的操作模式，就是“打开-读-写-关闭”（open - read - write - close）的I/O模式。当TCP/IP协议被集成到UNIX内核中的时候，相当于在UNIX系统中引入了一种新型的I/O操作，就是应用程序通过网络协议栈来交换数据。在UNIX系统的实现中，套接字是完全与其他I/O集成在一起的。操作系统和应用程序都将套接字编程接口也看作一种输入/输出机制。但是，用户进程与网络协议的交互作用实际要比用户进程与传统的I/O设备相互作用要复杂得多。其次，使用套接字的应用程序必须说明许多细节。仅仅提供open、read、write、close四个过程远远不够。

为避免单个套接字函数参数过多，套接字编程接口的设计者定义了多个函数。在不同类型的套接字的实现中，使用不同的函数。

* 1. 面向连接的套接字编程
     1. 工作流程

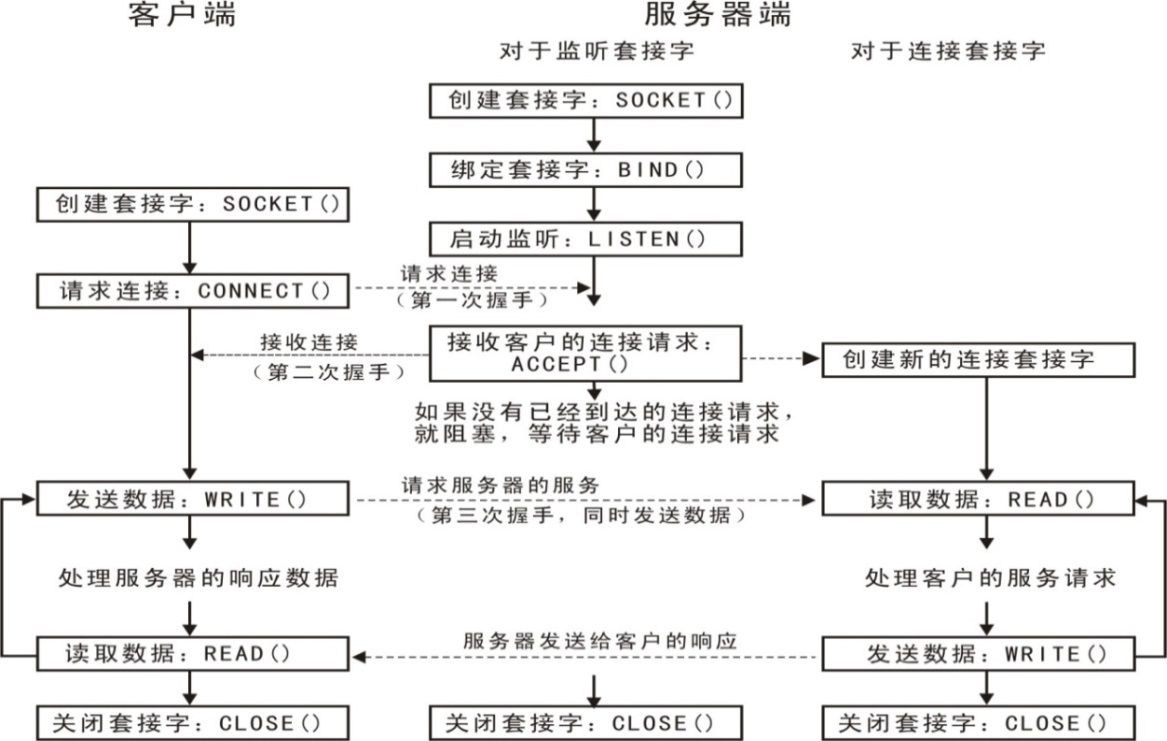


图3-1 流式套接字工作流程

* + 1. 关键函数讲解

（1）SOCKET函数

int SOCKET( int domain, int Type, int Protocol);

返回值： 若成功则为非负描述符，出错则为-1。

参数说明：

domain:   指明协议族，也称为协议域，是一个常值。

AF\_INET           IPv4 协议

AF\_INET6           IPv6 协议

AF\_LOCAL/AF\_UNIX Unix协议域

AF\_ROUTE               路由套接字

AF\_KEY                 密匙套接字

type：    指明套接字的类型。

SOCK\_STREAM           字节流套接字

SOCK\_DGRAM            数据报套接字

SOCK\_SEQPACKET        有序分组套接字

SOCK\_RAW              原始套接字

protocol: 指明协议类型。一般为0，以选择给定的domain和type组合的系统默认值。

IPPROTO\_TCP           TCP传输协议

IPPROTO\_UDP           UDP传输协议

IPPROTO\_SCTP          SCTP传输协议

函数描述： socket 函数在成功时返回一个小的非负整数值，与文件描述符类似，我们称它为套接字描述符，简称 sockfd。注意：这里只是指定了协议族（IPv4、IPv6或Unix）和套接字类型（字节流、数据报或原始套接字）。

（2）BIND函数

int bind(int sockfd, const struct sockaddr \*myaddr, socklen\_t addrlen);

返回值： 若成功则为0，出错则为-1。

参数说明：

sockfd:    socket 函数返回的套接字描述符。

myaddr、addrlen：指向一个套接字地址结构的指针和该结构的大小。

函数描述：将一个本地协议地址赋予一个套接字。bind 函数主要用于服务器端，用来指定本地主机的哪个网络地址（IP，可以是INADDR\_ANY，表示本地主机的任一网络接口）可以接受客户端的请求，和指定端口号（即开启的等待客户来连接的进程）。

（3）LISTEN函数

int LISTEN( int Sockfd, int Queuesize)

返回：若成功则为0，出错则为-1。

函数描述：

listen 函数仅由 TCP 服务器调用，它做两件事情。

(1)把一个未连接的套接字（主动）转换成一个被动套接字，指示内核应该接受指向该套接字的连接请求。

(2) Queuesize 参数规定了内核应该为相应套接字排队的最大连接数。其中内核始终为监听套接字维护两个 队列。

1)未完成连接队列，每个SYN分节对于其中一项：已由某个客户发出并到达服务器，而服务器正在等待待完成的TCP三路握手过程。这些套接字处于SYN\_RCVD状态。2)已完成连接队列每个已完成TCP三路握手过程的客户对应其中一项。这些套接字处于ESTABLISHED状态。backlog 就是这两个队列和的最大值。在三路握手完成之后，但在服务器调用 accept 之前到达的数据应由 TCP 服务器排队，最大数据量为相应已连接套接字的接收缓冲区的大小。

（4）ACCEPT函数

int ACCEPT(int Sockfd, struct sockaddr\* cliaddr, int\* addrlen);

返回值：若成功则为非负，已连接描述符和对端的IP和端口号，出错则为-1。

参数说明：

cliaddr、addrlen 用来返回已连接的对端进程（客户）的协议地址。调用前，我们将由 \*addrlen 所引用的整数值置为由cliaddr所指的套接字地址结构的长度，返回时，该整数值即为内核存放在该套接字地址机构内的确切字节数。

函数描述：返回新的套接字与客户端连接.如果 accept 调用成功，那么其返回值是由内核自动生成的一个全新描述符，代表着与所返回客户的TCP连接。在讨论 accept 函数时，我们称它的第一个参数为监听套接字描述符（由 socket 创建，随后用作bind 和 listen 的第一个参数的描述符），称它的返回值为已连接套接字描述符。区分这两个套接字非常重要。一个服务器通常仅仅创建一个监听套接字，它在服务器的生命期内一直存在。内核为每个服务器进程接受的客户连接创建一个已连接套接字（也就是说对于它的TCP三路握手过程已经完成）。当服务器完成对某个连接客户的服务时，相应的已连接套接字就要被关闭。

（5）CONNECT函数

int CONNECT( int Sockfd, struct sockaddr\* Service\_addr, int Addrlen);

 TCP 客户用 connect 函数来与 TCP 服务器建立连接。

返回值：若成功则为0，出错则为-1。

参数说明：

sockfd:由 socket 函数创建的套接字描述符。

servaddr、addrlen：指向一个套接字地址结构的指针和该结构的大小。套接字地址结构必须含有服务器的IP地址和端口号。

函数描述：客户在调用 connect 函数前并不一定得调用 bind 函数，如果需要的话，内核会确定源IP地址，并选择一个临时端口作为源端口。所以在客户进程中的套接字一般只需指明客户所要连接的服务器的IP跟端口号。如果是 TCP 套接字，调用 connect 函数将激发 TCP 的三路握手。而且仅在连接成功或出错时才返回。若 connect 调用失败则该套接字不再可用，必须关闭，不能对这样的套接字再次执行connect 函数。

（6）READ函数

int READ( int sockfd, void\* buffer, int len );

Read函数是负责从fd中读取内容，当读取成功时，read返回实际读取到的字节数。

如果返回值等于0，表示已经读取到文件的结束了。

如果返回值小于0，表示是读取错误。错误代码EINTR表示在写的时候出现了中断错误，错误代码EPIPE表示网络连接出现了问题。

（7）WRITE函数

int WRITE( int sockfd, void\* buffer, int len )

Write函数将buf中的nbytes字节内容写入到文件描述符中，成功返回写的字节数。

如果write的返回值大于0，表示写了部分数据或者是全部的数据。需要注意的是，写数据循环过程中的buf参数和nbytes参数是需要程序控制的。也就是说，写函数是不负责将全部数据写完之后再返回的，有可能中途就返回了！

如果返回值小于0，此时出错了，需要根据错误类型进行相应的处理。错误代码是EINTR---表示在写的时候出现了中断错误，错误代码是EPIPE表示网络连接出现了问题。

有了上面的两个函数，我们就可以向客户端或者是服务器端进行数据传输了！比如我要传送一个结构体，可以使用下面的方法：

客户端向服务器：

Struct student stu;

Write(sock,(void \*)&stu,sizeof(struct student));

服务器读：

Char buffer[sizeof(struct student)];

Struct \*my\_student;

Read(sock,(void \*)buffer,sizeof(struct student));

My\_student=(struct student)buffer;

在网络上传递数据时，我们一般把数据转换为char类型，接收的时候也一样的。

（8）SEND函数和RECV函数

Recv函数和Send函数提供了read和write函数一样的功能，不同的是他们提供了四个参数。

Int recv(int fd,void \*buf,int len,int flags)

Int send(int fd,void \*buf,int len,int flags)

前面的三个参数和read、write函数是一样的。第四个参数可以是0或者是以下组合：

MSG\_DONTROUTE：不查找路由表。是send函数使用的标志，这个标志告诉IP，目的主机在本地网络上，没有必要查找表，这个标志一般用在网络诊断和路由程序里面。

MSG\_OOB：接受或者发生带外数据。表示可以接收和发送带外数据。

MSG\_PEEK：查看数据，并不从系统缓冲区移走数据。是recv函数使用的标志，表示只是从系统缓冲区中读取内容，而不清楚系统缓冲区的内容。这样在下次读取的时候，依然是一样的内容，一般在有多个进程读写数据的时候使用这个标志。

MSG\_WAITALL：等待所有数据。是recv函数的使用标志，表示等到所有的信息到达时才返回，使用这个标志的时候，recv返回一直阻塞，直到指定的条件满足时，或者是发生了错误。

（9）CLOSE函数

int CLOSE( int sockfd );

* + 1. 面向连接的SOCKET示例

**实例的功能**

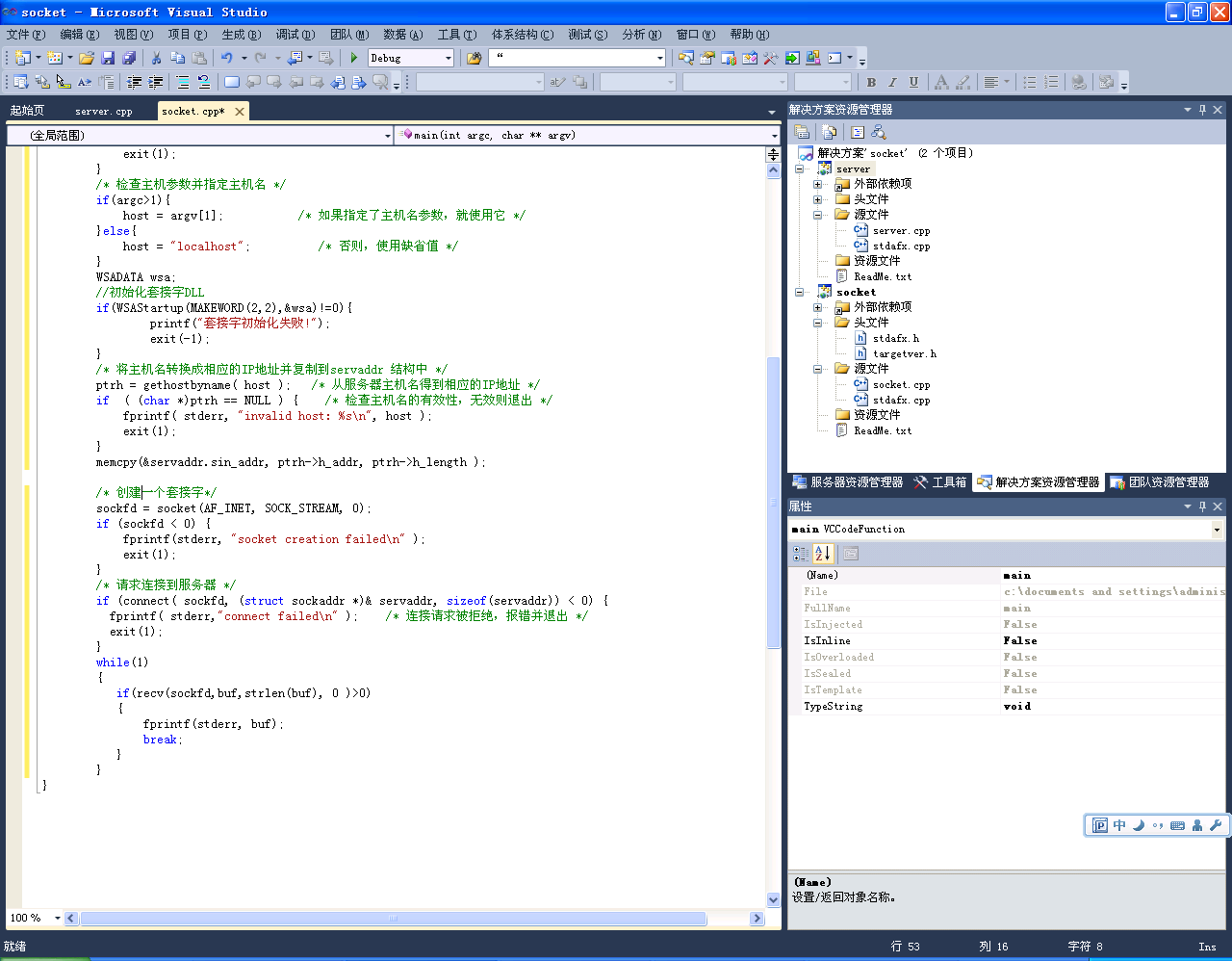
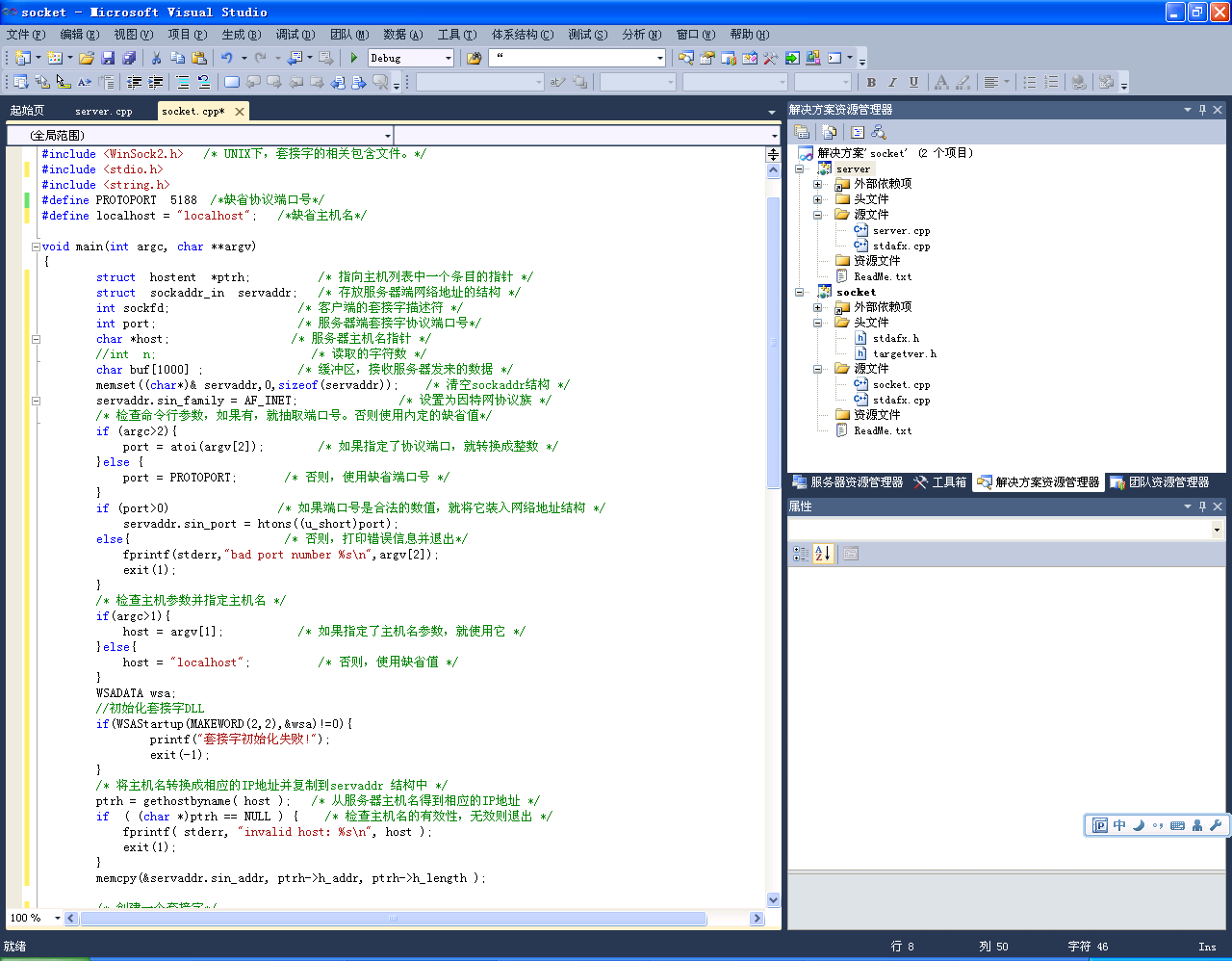
（1）服务器对来访的客户计数，并向客户报告这个计数值。

（2）客户建立与服务器的一个连接并等待它的输出。

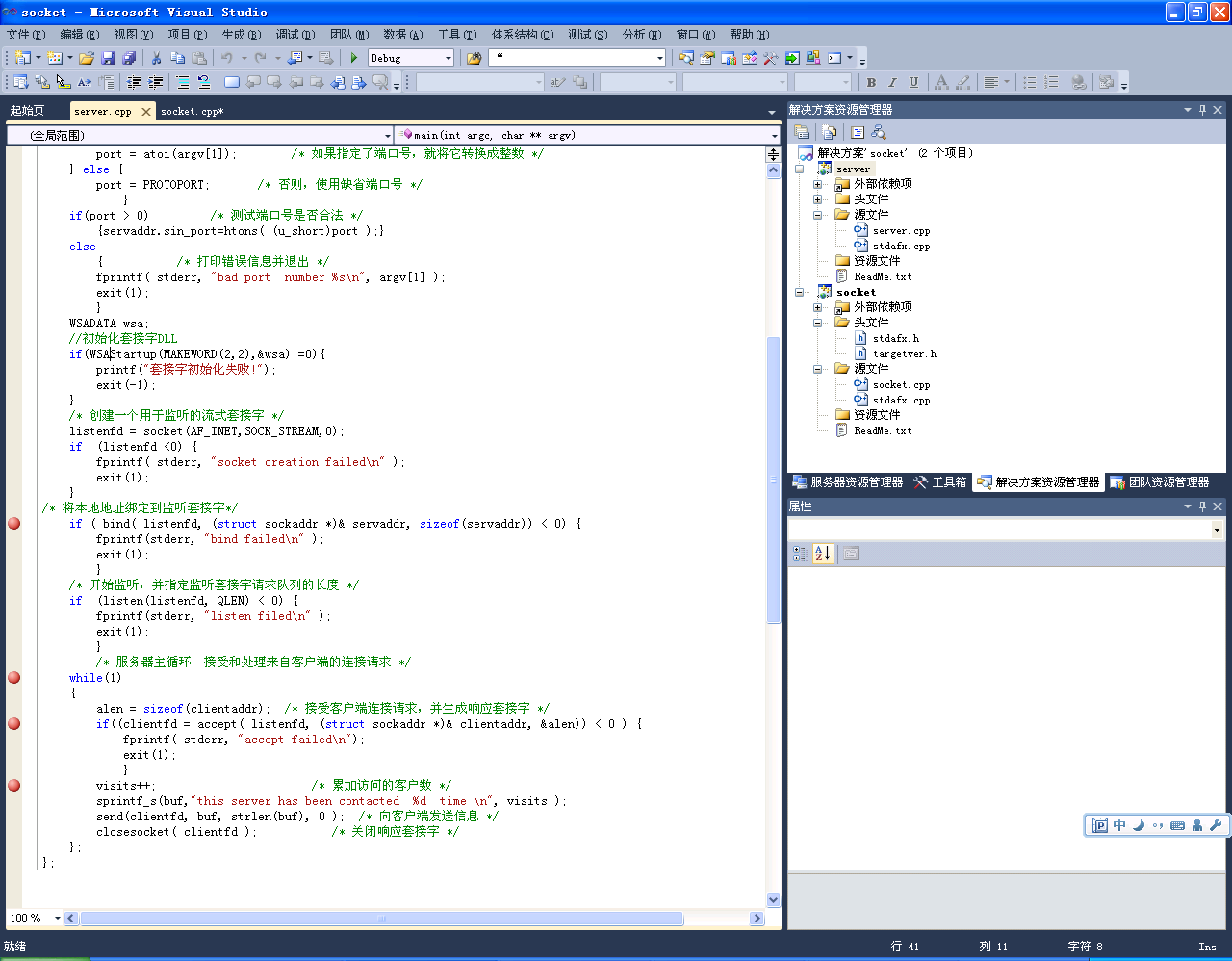
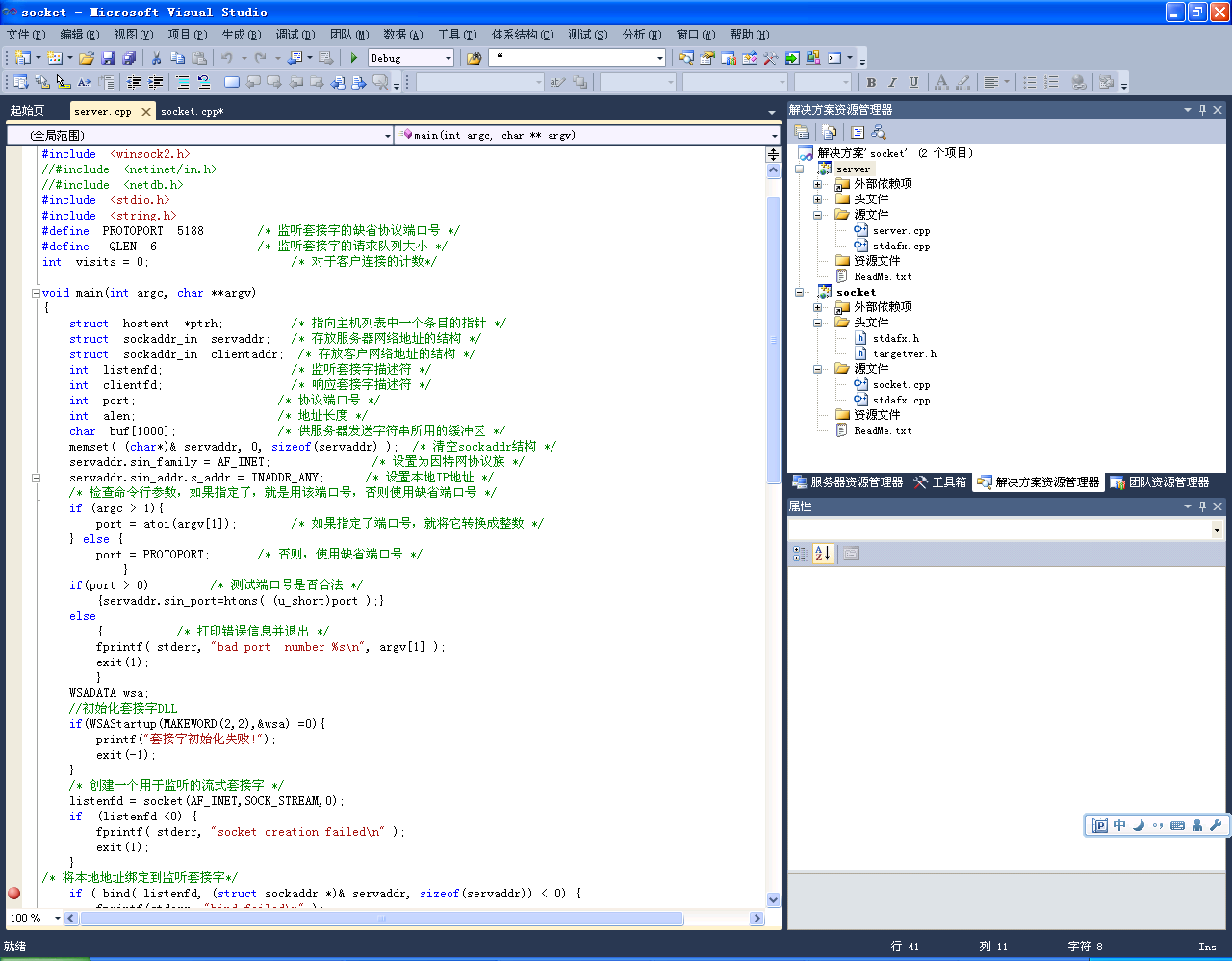
（3）每当连接请求到达时，服务器生成一个可打印的ASCII串信息，将它在连接上发回，然后关闭连接。

（4）客户显示收到的信息，然后退出。

TCP\_Client.cpp



TCP\_Server.cpp



* 1. 关于阻塞
     1. 阻塞的基本概念

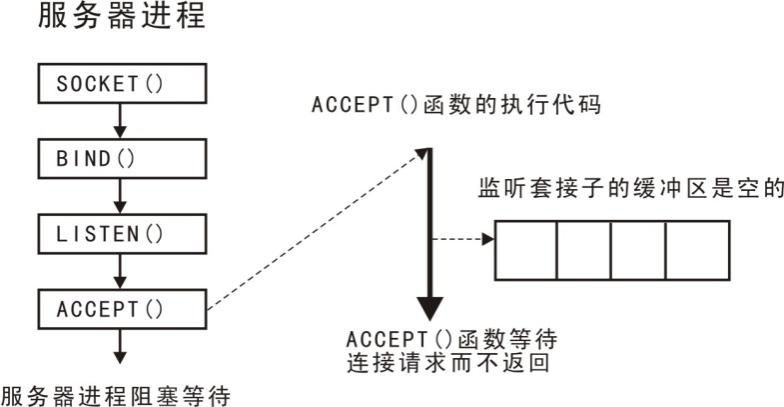


图3-2 阻塞模式

阻塞是指一个进程执行了一个函数或者系统调用，该函数由于某种原因不能立即完成，因而不能返回调用它的进程，导致进程受控于这个函数而处于等待的状态，进程的这种状态称为阻塞。

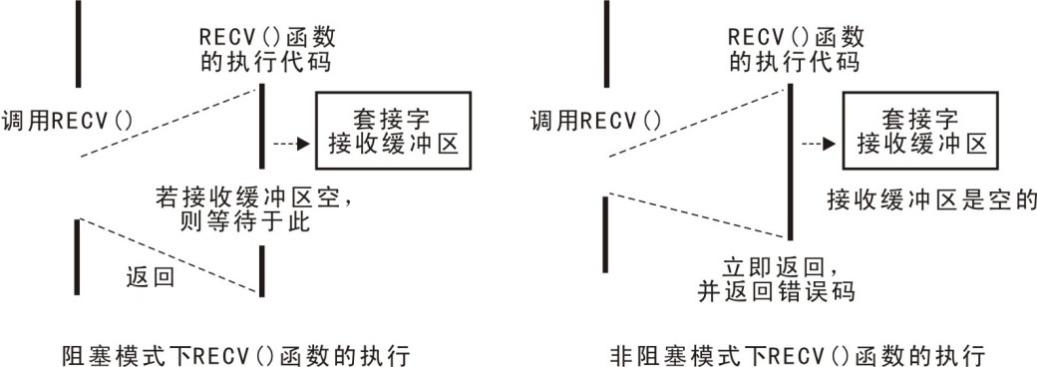


图3-3 阻塞模式和非阻塞模式的区别

* + 1. 引发阻塞的函数

在Berkeley套接字网络编程接口的模型中，套接字的默认行为是阻塞的，具体地说，在一定情况下，有多个操作套接字的系统调用会引起进程阻塞。

（1）ACCEPT()

（2）READ()、RECV()和READFORM()

（3）WRITE()、SEND()和SENDTO()

（4）CONNECT()

（5）SELECT()

（6）CLOSESOCKET()

* + 1. 多线程程序

采用阻塞工作模式的单进程服务器是不能很好地同时为多个客户服务的。

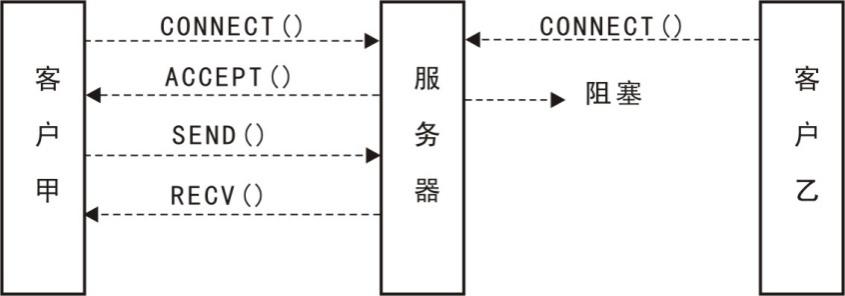


图3-4 阻塞模式的缺点

并发性是客户／服务器交互模式的基础，并发允许多个客户获得同一种服务，而不必等待服务器完成对上一个请求的处理。这样才能很好地同时为多个客户提供服务。

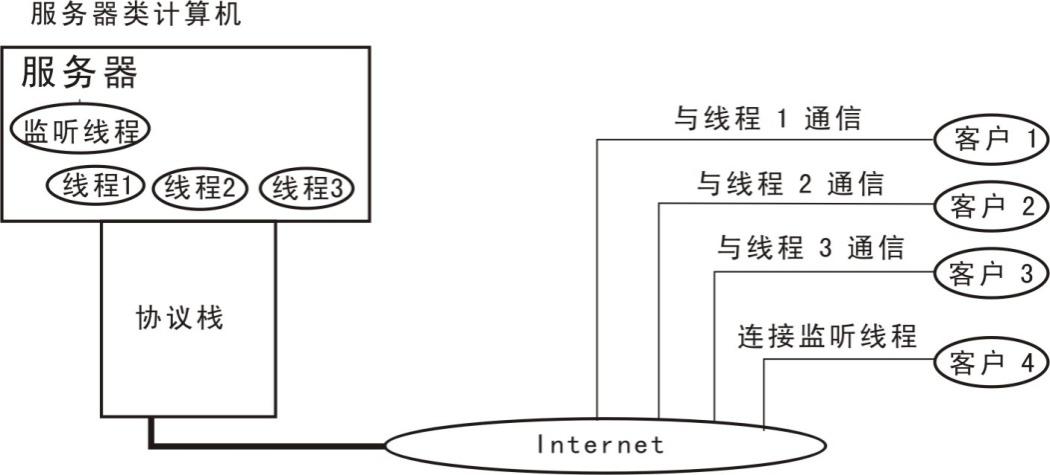


图3-5 阻塞模式的应用方案

利用UNIX操作系统的FORK()系统调用，编制多进程并发执行的服务器程序。可以创建子进程。对于每一个客户端，用一个专门的进程为它服务，通过进程的并发执行，来实现对多个客户的并发服务。基本的编程框架是：

父进程代码

If ((pid = FORK()) == 0) {

…….

子进程代码

........

} else if (pid<0) {

报错信息

}

父进程代码

Win32多线程设计，参见：

http://baike.baidu.com/link?url=mfEJokKKSoOXcRQ4pZmWTid304whz0jrQDNYrWLAOzxkQ8W8TPtNPMPasJNBMaAlLP938JOjNIgtN2jBTgAzG\_

* 1. 面向无连接的套接字编程
     1. 工作流程

使用数据报套接字开发网络应用程序，其基本的过程如下图所示。

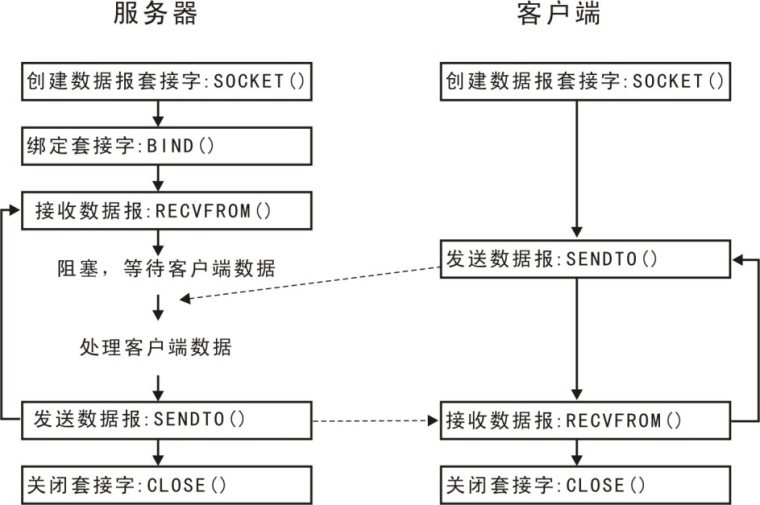


图3-6 数据报套接字工作流程

专用的函数调用：

* 1. 发送数据报SENDTO

int sendto(

\_\_in SOCKET s,

\_\_in const char \*buf,

\_\_in int len,

\_\_in int flags,

\_\_in const struct sockaddr \*to,

\_\_in int tolen

);

返回值：

若无错误发生，send()返回所发送数据的总数（请注意这个数字可能小于len中所规定的大小）。否则的话，返回SOCKET\_ERROR错误，应用程序可通过WSAGetLastError()获取相应错误代码。

参数说明：

s：一个标识套接口的描述字。

buf：包含待发送数据的缓冲区。

len：buf缓冲区中数据的长度。

flags：调用方式标志位。

to：（可选）指针，指向目的套接口的地址。

tolen：to所指地址的长度。

注释：sendto()函数主要用于SOCK\_DGRAM类型套接口向to参数指定端的套接口发送数据报。必需注意发送数据长度不应超过通讯子网的IP包最大长度。如果数据太长无法自动通过下层协议，则返回WSAEMSGSIZE错误，数据不会被发送。

为了发送广播数据（仅适用于SOCK\_DGRAM），in参数所含地址应该把特定的IP地址INADDR\_BROADCAST（winsock.h中有定义）和终端地址结合起来构造。通常建议一个广播数据报的大小不要大到以致产生碎片，也就是说数据报的数据部分（包括头）不超过512字节。

如果传送系统的缓冲区空间不够保存需传送的数据，除非套接口处于非阻塞I/O方式，否则sendto()将阻塞。

在相关套接口的选项之上，还可通过标志位flag来影响函数的执行方式。标志位取值为MSG\_DONTROUTE:指明数据不选径，MSG\_OOB:发送带外数据

（2）接收数据报 RECVFROM

int recvfrom(

\_\_in SOCKET s,

\_\_out char \*buf,

\_\_in int len,

\_\_in int flags,

\_\_out struct sockaddr \*from,

\_\_inout opt int \*fromlen

);

返回值：

若无错误发生，recvfrom()返回读入的字节数。如果连接已中止，返回0。否则的话，返回SOCKET\_ERROR错误。

参数说明：

s：标识一个已连接套接口的描述字。

buf：接收数据缓冲区。

len：缓冲区长度。

flags：调用操作方式。

from：（可选）指针，指向装有源地址的缓冲区。

fromlen：（可选）指针，指向from缓冲区长度值。

注释：

对于数据报类套接口，队列中第一个数据报中的数据被解包，但最多不超过缓冲区的大小。如果数据报大于缓冲区，那么缓冲区中只有数据报的前面部分，其他的数据都丢失了，并且recvfrom()函数返回错误。

若from非零，且套接口为SOCK\_DGRAM类型，则发送数据源的地址被复制到相应的sockaddr结构中。fromlen所指向的值初始化时为这个结构的大小，当调用返回时按实际地址所占的空间进行修改。

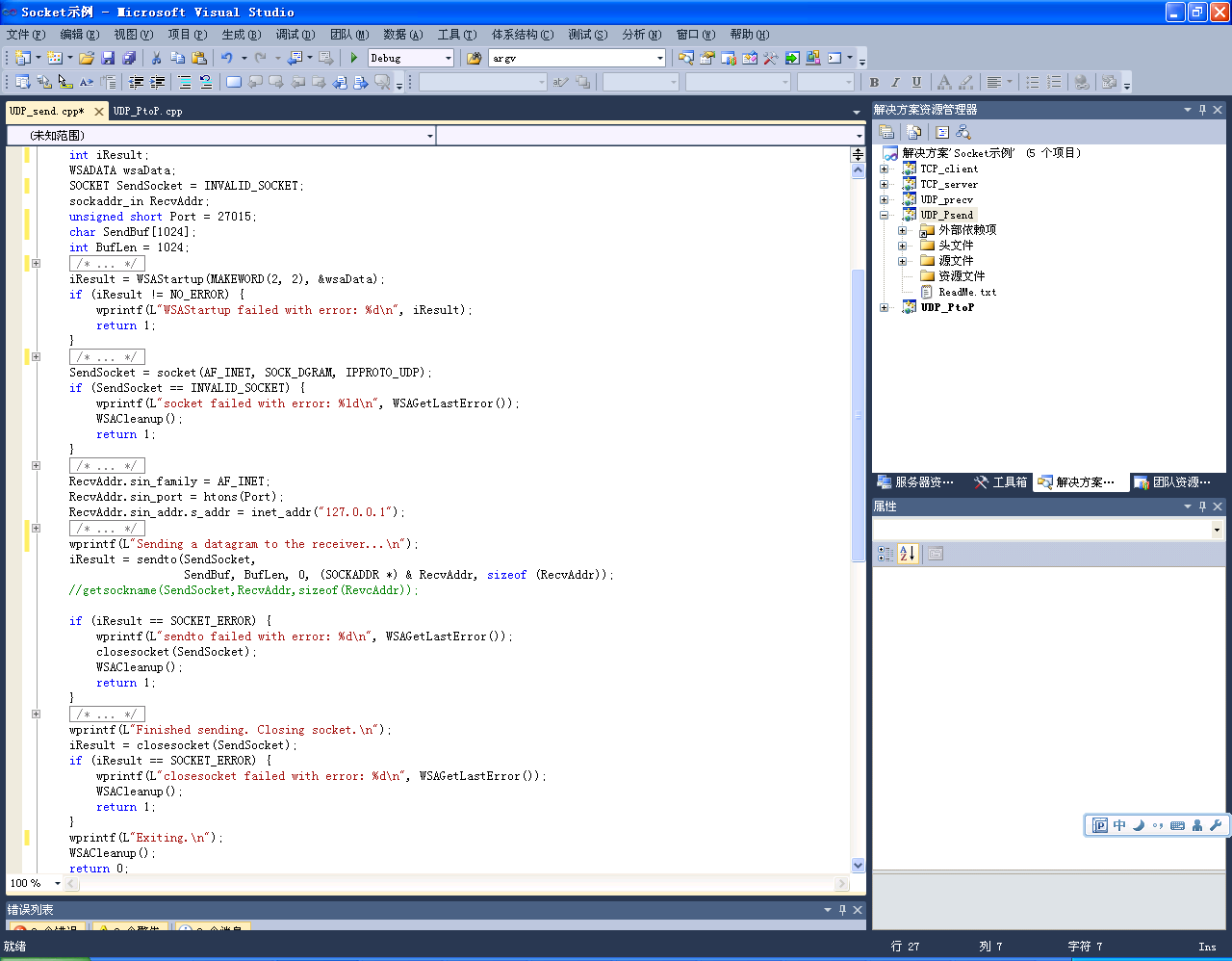
如果没有数据待读，那么除非是非阻塞模式，不然的话套接口将一直等待数据的到来，此时将返回SOCKET\_ERROR错误。

在套接口的所设选项之上，还可用标志位flag来影响函数的执行方式。也就是说，本函数的语义既取决于套接口选项，也取决于标志位参数。标志位为MSG\_PEEK： 查看当前数据。数据将被复制到缓冲区中，但并不从输入队列中删除，MSG\_OOB 处理带外数据。

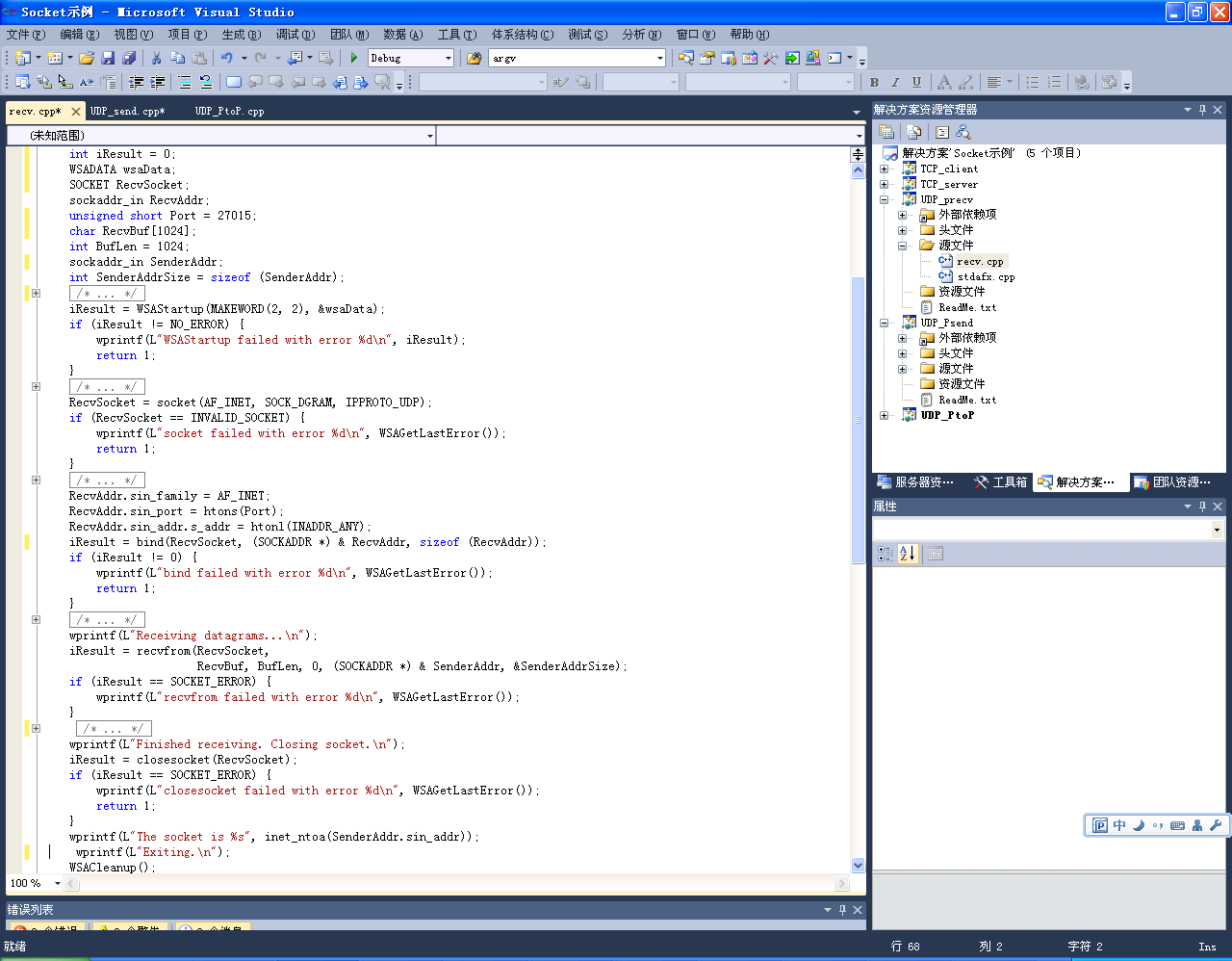
* + 1. 面向无连接的SOCKET示例

示例功能：UDP\_Psend程序发送消息，UDP\_Precv程序等待接收，接收到消息，退出，否则等待。

UDP\_Psend.cpp //发送程序



UDP\_Precv.cpp



* 1. RAW socket套接字编程

基于IP层的直接传输。

Ping.cpp

* 1. 上机实验

上机实验 1：下面三个实验，选1完成。

（1）编写ping程序，raw socket；

（2）无连接的控制台聊天程序；

（3）面向连接的多用户服务器；

1. B/S程序设计概述
   1. B/S架构应用程序工作原理
      1. 基本概念

采用B/S模式开发的应用程序，也称为Web应用程序。通俗来讲，Web程序就是一种通过浏览器发送HTTP请求，由服务器响应后返回HTML文本供浏览器进行显示的应用程序。



图4-1 Web应用程序的基本组成

一个完整的Web应用程序的基本组成部分包括：

1. 解析网页并显示网页给用户浏览的客户端浏览器；
2. 在客户端和Web服务器端提供通讯的HTTP网络协议；
3. 提供服务的Web服务器，---供浏览器进行链接访问；
4. 由网页组成的Web应用程序，---存储在web服务器上；
5. 执行用户逻辑的应用服务器；---应用服务器软件
6. 可能还包括数据库服务器等。---数据库软件

工作过程：客户端发起http请求，浏览器与相应的服务器进行连接，传送http请求，web服务器接收后，进行响应，应用服务器完成相应功能的执行，将结果返回给浏览器。

那么，浏览器和服务器之间的通信采用什么方式呢？

* + 1. B/S架构程序的内部机理

Web应用程序也是Internet程序，因此，其通信也是通过Socket，但是，我们看不到Socket的建立和释放过程，这些过程都由浏览器和WWW服务器帮助我们完成了。下图展示了B/S架构程序的内部机理。

浏览器接收用户输入的URL地址

浏览器解析地址，得到**IP和端口号**

浏览器建立Stream\_Socket，与远程web服务器进行连接

web服务器响应请求，双方建立连接

web服务器返回默认主页的内容（HTML文件）

浏览器进行解析、展示

用户操作页面上的内容，通过超链接请求新的链接地址，或者通过Post提交内容到指定网页

web服务器接收请求，调用应用服务器、数据库服务器等完成请求，生成ＨＴＭＬ页面返回

图4-2 B/S架构程序执行的内部机理

Web服务是基于TCP协议的，下面的过程演示了在访问一个网址时，浏览器与服务器的连接过程。

演示重点：讲解连接也是通过socket方式，深入理解B/S和C/S编程方式的区别与联系。

网页的访问过程。[www.baidu.com](http://www.baidu.com/)，其包交换过程如图4-3所示。NO.45-47是进行TCP的三次握手，其发起也是由Socket建立请求的。NO.48发送了一个HTTP请求，其本质仍是一个TCP包，NO.51收到后首先自动应答，表明通信正确，然后对请求响应后再发一个内容包NO.52，告诉请求者，一切都好，你继续吧。其中的17953就是端口号。

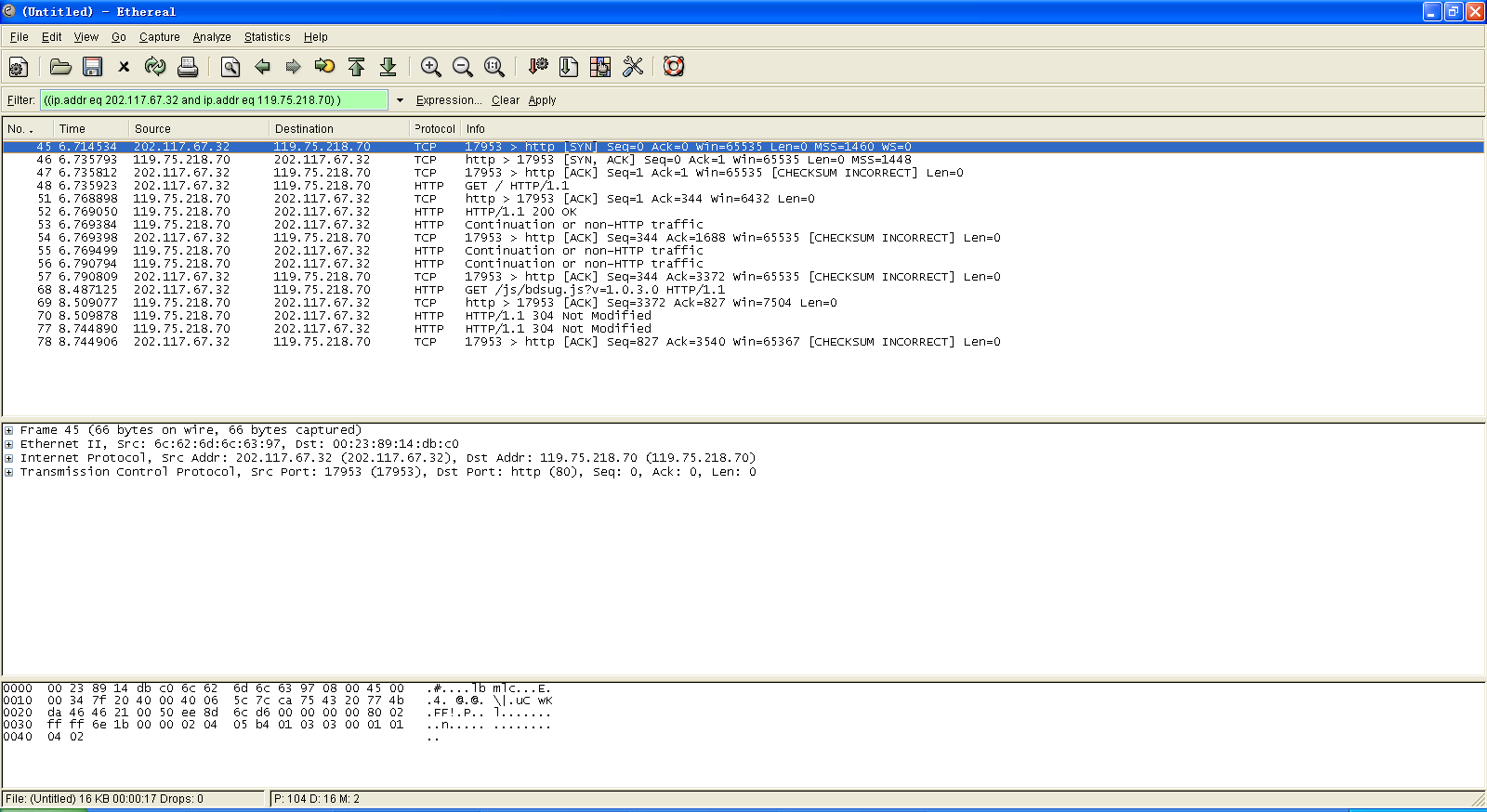


图4-3 访问[www.baidu.com](http://www.baidu.com/)的包交换过程

在同一窗口，访问[www.chd.edu.cn](http://www.chd.edu.cn/)。由于IP地址不同了，TCP的连接重新建立。端口号发生了变化，成了17966了，见图4-4。

思考：如果重新访问上一个网页，连接的情况会怎样呢？

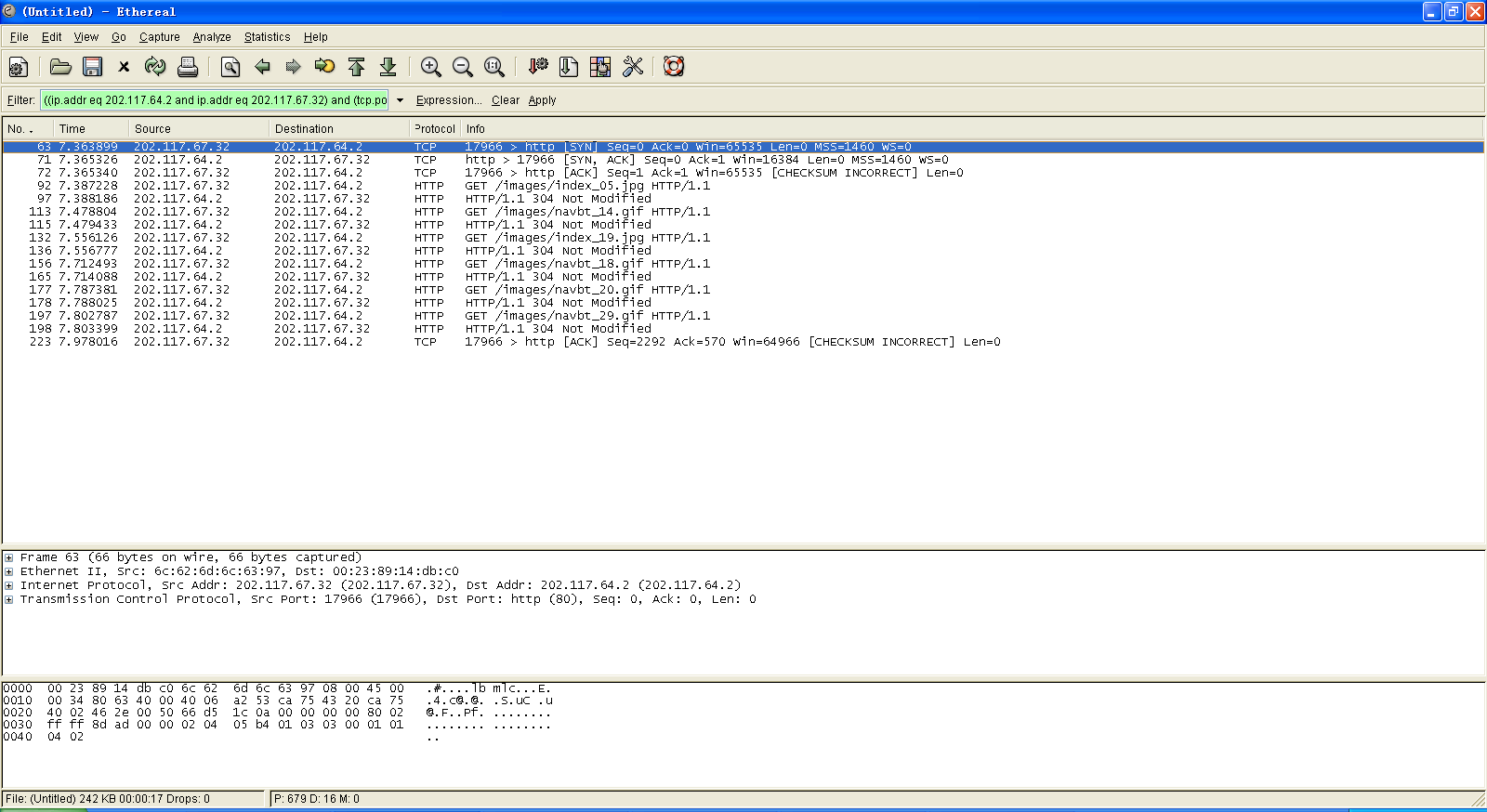


图4-4 在同一窗口访问其他网站的包交换过程

在该网页上点击xyh.chd.edu.cn可以看到，由于IP地址不同了，TCP的连接也重新建立。

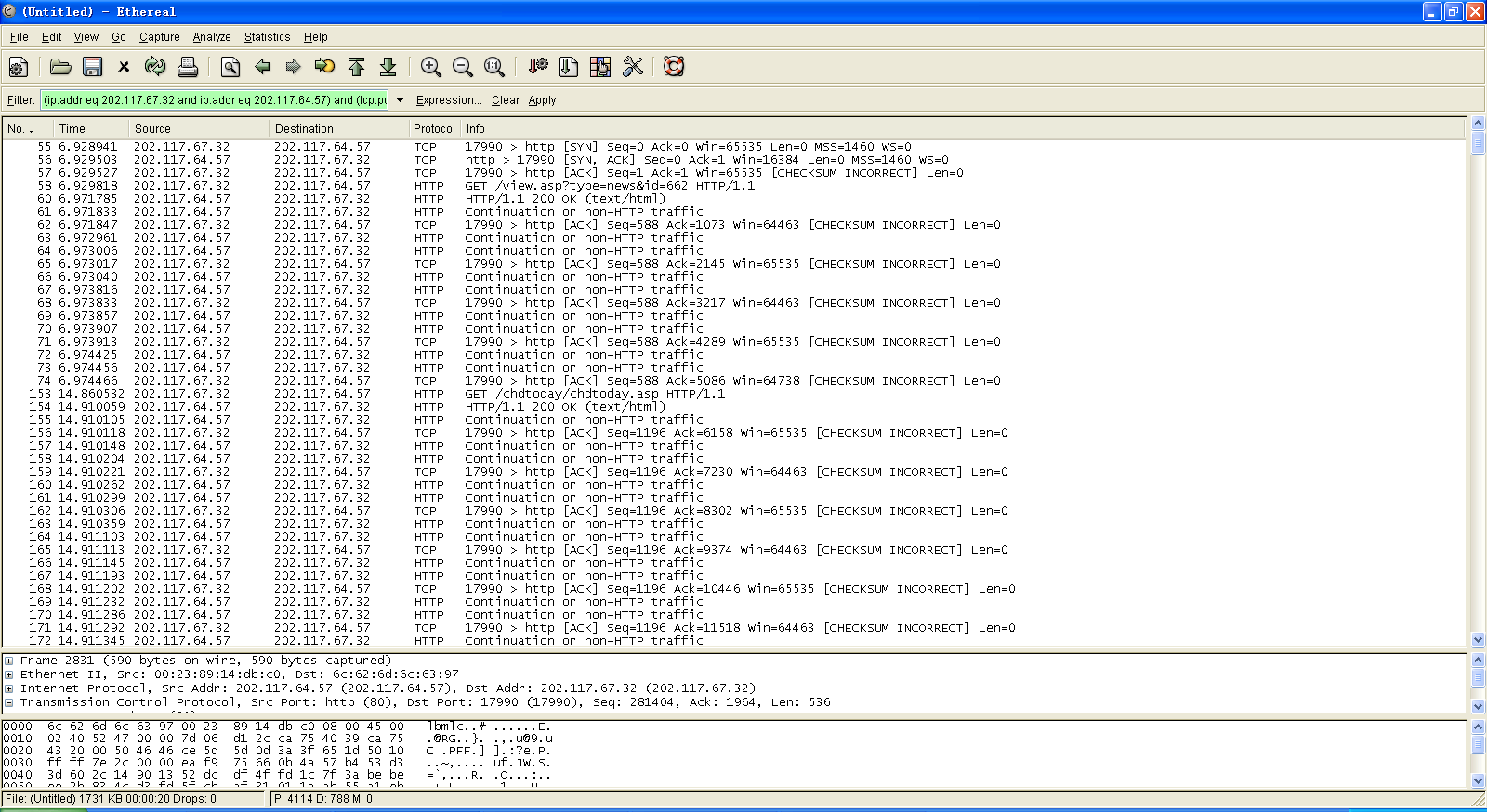


图4-5 在该网页上点击其他网站的网址

从上述过程可以看出，B/S架构的应用程序建立Socket连接的过程由浏览器做了，通信内容的格式也自动产生了（HTTP协议）。实际上是简单了。但本质上，仍然是基于Socket的网络通信过程。

作业：实现一个简单的浏览器：接收用户输入的URL地址，建立与服务器的连接，接收返回的数据，并将数据直接输出。

* + 1. Internet程序通信方法小结

（1）通过第三章的学习，掌握了C/S架构程序的编程方法。

讨论一下QQ聊天程序、网络游戏、OUTLOOK 程序等基于互联网的程序的实现方法。

（2）B/S架构程序的原理。与C/S架构类似，只不过在C/S中，通信用的Socket需要自己定义，而且通信内容也需要自己定义，B/S中的通信内容被HTTP协议规定好了。

延伸思考和讨论：还有什么应用程序的实现原理不清楚？

问题：（1）FTP服务器如何实现的？

* 1. B/S架构应用程序开发的基础
* Web服务器及其部署；
* HTTP协议理解；
  + 1. Web服务器简介及架设
       1. Web服务器和应用服务器

Web服务器的作用是维护和客户端的连接，并响应HTTP请求。当Web服务器接收到一个HTTP请求(request)，会返回一个HTTP响应(response)，例如送回一个HTML页面。为了处理一个请求(request)，Web服务器可以响应(response)一个静态页面或图片，进行页面跳转(redirect)，或者把动态响应(dynamic response)的产生委托(delegate)给一些其它的程序例如CGI脚本、JSP(JavaServer Pages)脚本、servlets、ASP(Active Server Pages)脚本、服务器端(server-side)JavaScript或者一些其它的服务器端(server-side)技术。无论它们的目的如何，这些服务器端的程序通常产生一个HTML的响应来让浏览器可以浏览。

应用程序服务器是通过HTTP等多种协议来为应用程序提供商业逻辑(business logic)。商业逻辑就象你调用对象的一个方法(或过程语言中的一个函数)一样。在大多数情形下，应用程序服务器是通过组件(component)的应用程序接口(API)把商业逻辑暴露(expose)(给客户端应用程序)的，例如基于J2EE(Java 2 Platform, Enterprise Edition)应用程序服务器的EJB(Enterprise JavaBean)组件模型。此外，应用程序服务器可以管理自己的资源，例如看大门的工作(gate-keeping duties)包括安全(security)、事务处理(transaction processing)、资源池(resource pooling) 和消息(messaging)。应用程序服务器还配置了多种可扩展(scalability)和容错(fault tolerance)技术。 来

现在大多数应用程序服务器也包含了Web服务器。但是开发者很少把应用程序服务器部署(deploy)成这样。相反，通常会把Web服务器独立配置，和应用程序服务器一前一后。这种功能的分离有助于提高性能，而且给最佳产品的选取留有余地。

* + - 1. 常用的Web服务器

在UNIX和LINUX平台下使用最广泛的免费HTTP服务器是W3C、 APACHE服务器等，而Windows平台NT/2000/2003使用IIS的WEB服务器。在选择使用WEB服务器应考虑的本身特性因素有：性能、安全性、日志和统计、虚拟主机、代理服务器、缓冲服务和集成应用程序等，下面介绍几种常用的WEB服务器。

* 1. Microsoft IIS

Microsoft的Web服务器产品为Internet Information Server (IIS)， IIS 是允许在公共Intranet或Internet上发布信息的Web服务器。IIS是目前最流行的Web服务器产品之一，很多著名的网站都是建立在IIS的平台上。IIS提供了一个图形界面的管理工具，称为 Internet服务管理器，可用于监视配置和控制Internet服务。

IIS是一种Web服务组件，其中包括Web服务器、FTP服务器、NNTP服务器和SMTP服务器，分别用于网页浏览、文件传输、新闻服务和邮件发送等方面，它使得在网络(包括互联网和局域网)上发布信息成了一件很容易的事。它提供ISAPI(Intranet Server API)作为扩展Web服务器功能的编程接口;同时，它还提供一个Internet数据库连接器，可以实现对数据库的查询和更新。

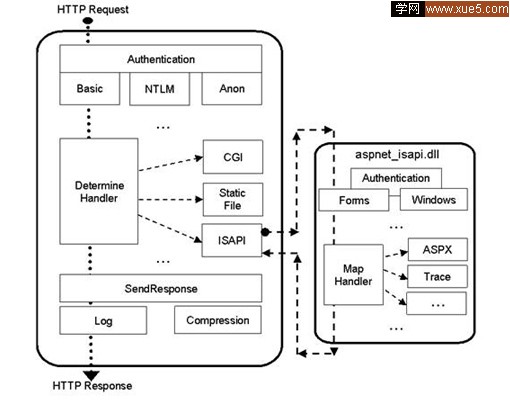


图4-6 IIS服务器的工作流程

官方网站：http://www.microsoft.com

* 1. **Apache HTTP Server**

Apache是世界使用排名第一的Web服务器软件。它可以运行在几乎所有广泛使用的计算机平台上。Apache取自“apatchy server”的读音，意思是充满补丁的服务器，因为它是自由软件，所以不断有人来为它开发新的功能、新的特性、修改原来的缺陷。Apache的诞生极富有戏剧性。当NCSA WWW服务器项目停顿后，那些使用NCSA WWW服务器的人们开始交换他们用于该服务器的补丁程序，他们也很快认识到成立管理这些补丁程序的论坛是必要的。就这样，诞生了Apache Group，后来这个团体在NCSA的基础上创建了Apache。

Apache的特点是简单、速度快、性能稳定，并可做代理服务器来使用。本来它只用于小型或试验Internet网络，后来逐步扩充到各种Unix系统中，尤其对Linux的支持相当完美。Apache有多种产品，可以支持SSL技术，支持多个虚拟主机。Apache是以进程为基础的结构，进程要比线程消耗更多的系统开支，不太适合于多处理器环境，因此，在一个Apache Web站点扩容时，通常是增加服务器或扩充群集节点而不是增加处理器。

到目前为止Apache仍然是世界上用的最多的Web服务器，市场占有率达60%左右。世界上很多著名的网站如Amazon、Yahoo!、W3 Consortium、Financial Times等都是Apache的产物，它的成功之处主要在于它的源代码开放、有一支开放的开发队伍、支持跨平台的应用（可以运行在几乎所有的Unix、Windows、Linux系统平台上）以及它的可移植性等方面。

* + - 1. 常见的应用服务器

1. **IIS7**

[IIS](http://server.zdnet.com.cn/files/list-0-0-184273-1-1.htm) 7 完全整合了.NET，架构的处理顺序有了很大的不同（如下图），最主要的原因就是 ASP.NET 从 [IIS](http://server.zdnet.com.cn/files/list-0-0-184273-1-1.htm) 插件（ISAPI extension）的角色，进入了 [IIS](http://server.zdnet.com.cn/files/list-0-0-184273-1-1.htm) 核心，而且也能以 ASP.NET 模块负责处理 [IIS](http://server.zdnet.com.cn/files/list-0-0-184273-1-1.htm) 7 的诸多类型要求。这些 ASP.NET 模块不只能处理 ASP.NET 网页程序，也能处理其他如 ASP 程序、PHP 程序或静态 HTML 网页，也因为 ASP.NET 的诸多功能已经成为 [IIS](http://server.zdnet.com.cn/files/list-0-0-184273-1-1.htm) 7 的一部份，因此 ASP 程序、PHP 程序或静态 HTML 网页等类型的要求，也能使用像是Forms认证（Forms Authentication）或输出缓存（Output Cache）等 ASP.NET 2.0 的功能。也因为 [IIS](http://server.zdnet.com.cn/files/list-0-0-184273-1-1.htm) 7 允许自行以 ASP.NET API 开发并加入模块，因此 ASP.NET 网页开发人员将更容易扩充 [IIS](http://server.zdnet.com.cn/files/list-0-0-184273-1-1.htm) 7 和网站应用程序的功能，甚至能自行以 .NET 编写管理 [IIS](http://server.zdnet.com.cn/files/list-0-0-184273-1-1.htm) 7 的程序。

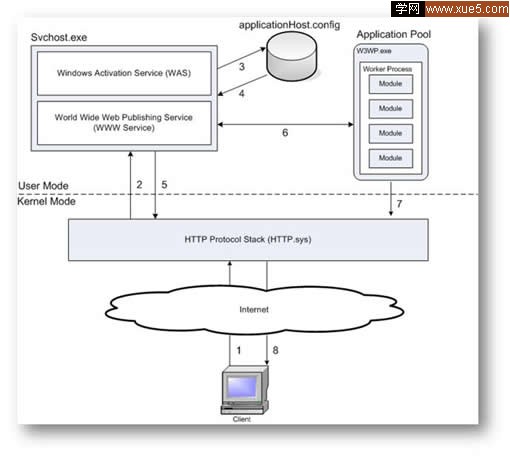


图 4-8 [IIS](http://server.zdnet.com.cn/files/list-0-0-184273-1-1.htm)7 站点启动并处理请求的步骤

上图的8个步骤分别如下：

* 当客户端浏览器开始HTTP 请求一个WEB 服务器的资源时，HTTP.sys 拦截到这个请求。
* HTTP.sys contacts WAS to obtain information from the configuration store.
* WAS 向配置存储中心请求配置信息。applicationHost.config。
* WWW 服务接受到配置信息，配置信息指类似应用程序池配置信息，站点配置信息等等。
* WWW 服务使用配置信息去配置 HTTP.sys 处理策略。
* WAS starts a worker process for the application pool to which the request was made.
* The worker process processes the request and returns a response to HTTP.sys.
* 客户端接受到处理结果信息。

1. **Apache Tomcat**

Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies.

Tomcat是一个开放源代码、运行servlet和JSP Web应用软件的基于Java的Web应用软件容器。Tomcat Server是根据servlet和JSP规范进行执行的，因此我们就可以说Tomcat Server也实行了Apache-Jakarta规范且比绝大多数商业应用软件服务器要好。

Tomcat是Java Servlet 2.2和JavaServer Pages 1.1技术的标准实现，是基于Apache许可证下开发的自由软件。Tomcat是完全重写的Servlet API 2.2和JSP 1.1兼容的Servlet/JSP容器。Tomcat使用了JServ的一些代码，特别是Apache服务适配器。随着Catalina Servlet引擎的出现，Tomcat第四版的性能得到提升，使得它成为一个值得考虑的Servlet/JSP容器，因此目前许多WEB服务器都是采用Tomcat。

Apache是普通服务器，本身只支持html即普通网页。不过可以通过插件支持php,还可以与Tomcat连通(单向Apache连接Tomcat,就是说通过Apache可以访问Tomcat资源。反之不然)。Apache只支持静态网页，但像asp,php,cgi,jsp等动态网页就需要Tomcat来处理。 Apache和Tomcat整合使用：如果客户端请求的是静态页面，则只需要Apache服务器响应请求；如果客户端请求动态页面，则是Tomcat服务器响应请求；因为jsp是服务器端解释代码的，这样整合就可以减少Tomcat的服务开销 。Apache和Tomcat是独立的，在同一台服务器上可以集成。

1. **IBM WebSphere**

WebSphere Application Server 是 一 种功能完善、开放的Web应用程序服务器，是IBM电子商务计划的核心部分，它是基于 Java 的应用环境，用于建立、部署和管理 Internet 和 Intranet Web 应用程序。 这一整套产品进行了扩展，以适应 Web 应用程序服务器的需要，范围从简单到高级直到企业级。

WebSphere 针对以 Web 为中心的开发人员，他们都是在基本 HTTP服务器和 CGI 编程技术上成长起来的。IBM 将提供 WebSphere 产品系列，通过提供综合资源、可重复使用的组件、功能强大并易于使用的工具、以及支持 HTTP 和 IIOP 通信的可伸缩运行时环境，来帮助这些用户从简单的 Web 应用程序转移到电子商务世界。

官方网站：http://www.ibm.com

下载地址：http://www.ibm.com/developerworks/cn/downloads/ws/wasce/

1. **BEA WebLogic**

BEA WebLogic Server 是一种多功能、基于标准的web应用服务器，为企业构建自己的应用提供了坚实的基础。各种应用开发、部署所有关键性的任务，无论是集成各种系统和数据库，还是提交服务、跨 Internet 协作，起始点都是 BEA WebLogic Server。由于它具有全面的功能、对开放标准的遵从性、多层架构、支持基于组件的开发，基于 Internet 的企业都选择它来开发、部署最佳的应用。

BEA WebLogic Server 在使应用服务器成为企业应用架构的基础方面继续处于领先地位。BEA WebLogic Server 为构建集成化的企业级应用提供了稳固的基础，它们以 Internet 的容量和速度，在连网的企业之间共享信息、提交服务，实现协作自动化。

官方网站：http://www.bea.com

下载地址：http://www.oracle.com/bea/index.html

* + - 1. Web服务器配置实验演示

演示实验 1：演示IIS的配置。

1. 添加IIS组件；
2. 设置，对IIS进行配置；
3. 访问，添加一个html页面，在客户端进行访问。

上机实验 2：对上述内容两人一组进行实验。

* + 1. HTTP协议

HTTP协议是用于Web服务器和浏览器之间的通信协议，它是一种请求/响应式的协议。基于HTTP协议的传输可分为四个过程：

* + 建立连接
  + 发送请求信息
  + 发送响应信息
  + 关闭连接

HTTP的请求命令：

表4-1 HTTP协议命令

|  |  |
| --- | --- |
| 方法 | 描述 |
| GET | 请求指定的文档 |
| HEAD | 仅请求指定文档的文档头信息 |
| POST | 请求服务器接收指定文档作为可执行的信息 |
| PUT | 用从客户端传送的数据取代指定文档中的内容 |
| DELETE | 请求服务器删除指定的页面 |
| OPTIONS | 允许客户端查看服务器的性能 |
| TRACE | 允许客户端查看消息回收过程（用于测试） |

请求例子：

一个HTTP请求的例子：

GET /servlet/default.jsp HTTP/1.1

Accept: text/plain; text/html

Accept-Language: en-gb

Connection: Keep-Alive

Host: localhost

Referer: http://localhost/ch8/SendDetails.htm

User-Agent: Mozilla/4.0 (compatible; MSIE 4.01; Windows 98)

Content-Length: 33

Content-Type: application/x-www-form-urlencoded

Accept-Encoding: gzip，deflate

userName=JavaJava&userID=javaID

HTTP响应报文与HTTP请求报文的结构很相似，主要区别在于报头部分，响应报文没有请求行，取而代之的是状态行，响应头中的字段也不相同

表4-2 HTTP响应报文

|  |  |  |
| --- | --- | --- |
| 状态码 | 类型 | 描述 |
| 1xx | 信息 | 表示服务器已收到请求，正在处理 |
| 2xx | 成功 | 表示客户端发送的请求已被服务器接受，并成功处理 |
| 3xx | 重定向 | 表示客户端必须提供更多的信息以从服务器获得成功响应 |
| 4xx | 客户端错误 | 表示请求中包含了错误的信息 |
| 5xx | 服务器错误 | 表示服务器在处理请求时发生了内部错误 |

打开一个页面，理解请求的例子：



图4-9 理解HTTP请求

最常见的两种请求：post，get。在两种请求模式下，浏览器都要和服务器通信，通信即要向服务器发送相应的请求信息，发送时，浏览器和服务器之间通过TCP连接，首先要建立和服务器相应端口的连接，然后发送相应的命令，接收返回信息。终止连接。

通常情况下是，浏览器（客户端）与服务器建立连接，然后向服务器请求资源（如网页），服务器收到请求后响应浏览器，把数据发送给浏览器。持续连接一直维持，直到浏览器通知服务器关闭连接为止。

延伸思考：不通知怎么办？

* + 1. HTML简介

超文本标记语言(HTML)是所有浏览器都可以理解的标准语言，它是一种描述文档结构的语言，而不能描述实际的表现形式。用HTML语言写的页面是普通文本文档(ASCII)，不含任何与平台和程序相关的信息，可被任何文本编辑器读取。用HTML语言描述的文档，需要用浏览器观看才能显示出效果，如字体格式，嵌入网页的图片、动画，网页的背景音乐等等。

HTML语言使用描述性的标记符(称为标签)来指明文档的不同内容。标签是区分文本各个组成部分的分界符，用来把HTML文档划分成不同的逻辑部分。标签描述了文档的结构，它向浏览器提供该文档的格式化信息，以传送文档的外观特征。

大部分HTML标签的形式如下：

* <标签名> 信息 </标签名>
* 起始标签与结束标签非常相似，只是结束标签的标签名前面多了一个斜杠（/）
* 某些HTML元素只有起始标签而没有相应的结束标签

<HTML>

<HEAD>

<TITLE>Sample Page</TITLE>

</HEAD>

<BODY>

<H1>Hello World</H1>

<B>Bold Text</B><BR>

<I>Italics Text</I><BR>

<U>Underlined Text</U><BR>

<LI>List Item 1

<LI>List Item 2

<LI>List Item 3

<BR>单击此处访问微软公司的网站 ->

<A HREF="http://www.microsft.com/">Microsoft</A>

</BODY>

</HTML>

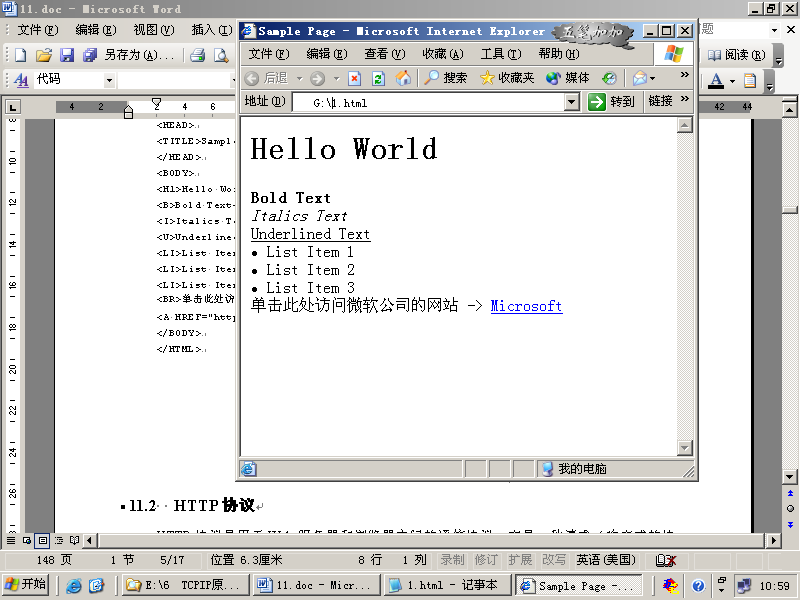


图4-10 HTML页面的源代码例子

*补充概念：URL（Uniform Resource Locator，统一资源定位符）*

HTML 不是一种编程语言，而是一种标记语言 (markup language)，标记语言是一套标记标签 (markup tag)，HTML 使用标记标签来描述网页。

静态网页的基本结构：

<html>

<head>

*……头文件内容*

</head>

<body>

*……网页内容*

</body>

</html>

1. B/S应用程序开发

从HTML来看，网页好像只能用来显示信息。那么，我们的程序放在什么地方呢？下面给大家分别解释静态和动态网页的开发技术。

* 1. 静态网页编程
     1. 基本知识

静态网页使用语言：HTML（超文本标记语言）。在网站设计中，纯粹HTML格式的网页通常被称为“静态网页”，早期的网站一般都是由静态[网页制作](http://edu.itbulo.com/web/)的。静态网页的文件通常以.htm、.html、.shtml、.xml等为后后缀的。

*概念区别：在HTML格式的网页上，也可以出现各种动态的效果，如.GIF格式的动画、FLASH、滚动字母等，这些“动态效果”只是视觉上的，与下面将要介绍的动态网页是不同的概念。静态网页可以简单理解为由服务器上下载下来完全由浏览器处理的网页，不能与服务器以及其他网页打交道。*

* + 1. 静态网页开发工具

Microsoft FrontPage，SharePoint design，Visual Studio，Macromedia DreamWeaver等

这些是辅助工具，许多人用写字板。无论什么工具，都是在写HTML文档。

* + 1. HTML标签

<http://www.w3school.com.cn/html/index.asp>

介绍：，讲解基本结构以及常用的标签。



演示实验 2：只包含基本网页元素的编程演示，使用visual studio演示htmltest.html的制作过程。像写文档一样简单。

HTML标签功能列表：<http://www.w3school.com.cn/tags/html_ref_byfunc.asp>

**上机实验 3：**根据标签功能列表，对每个标签进行练习，熟悉所有的标签功能，要求能够读懂简单的HTML文件。

|  |
| --- |
| <html xmlns="http://www.w3.org/1999/xhtml"> |
| <head> |
| <title>myHTMLtest测试程序</title> |
| </head> |
| <body style="height: 371px; width: 972px"> |
| <p align="center"><marquee width="798" height="20">Welcome, 交通信息工程系学生</marquee></p> |
| <table style="width: 82%; height: 298px; table-layout: fixed" border="2" bordercolor="#C0C0C0"> |
| <tr> |
| <td colspan="3" height="51"><strong>个人信息</strong></td> |
| </tr> |
| <tr> |
| <td align="center" width="33%" >姓名:</td> |
| <td width="33%">&nbsp;</td> |
| <td rowspan="5" width="384"><img id="mypic" border="0" src="images/j0335112.gif" width="264" height="265"></td> |
| </tr> |
| <tr><td align="center"width="33%"><div id="test">aaa: </div> </td> </tr> |
| </table> |
| <p style="text-align: center"></p> |
| <p align="center"> |
| <input id="button1" type="button" value="按一下开始执行对应的脚本"/> |
| <input id="button2" type="button" value="window" |
| onclick="return window\_onclick()" /></p> |
| <form id="form1"> |
| <select id="mySelect" name="D1"> |
| <option>Apple</option> |
| <option>Pear</option> |
| <option>Banana</option> |
| <option>Orange</option> |
| </select> |
| <input type="submit" onclick="insertOption()" value="Insert option" /> |
| </form> |
| </body> |
| </html> |

问题：静态网页能否执行程序逻辑？答案是能，要通过客户端脚本能够实现。

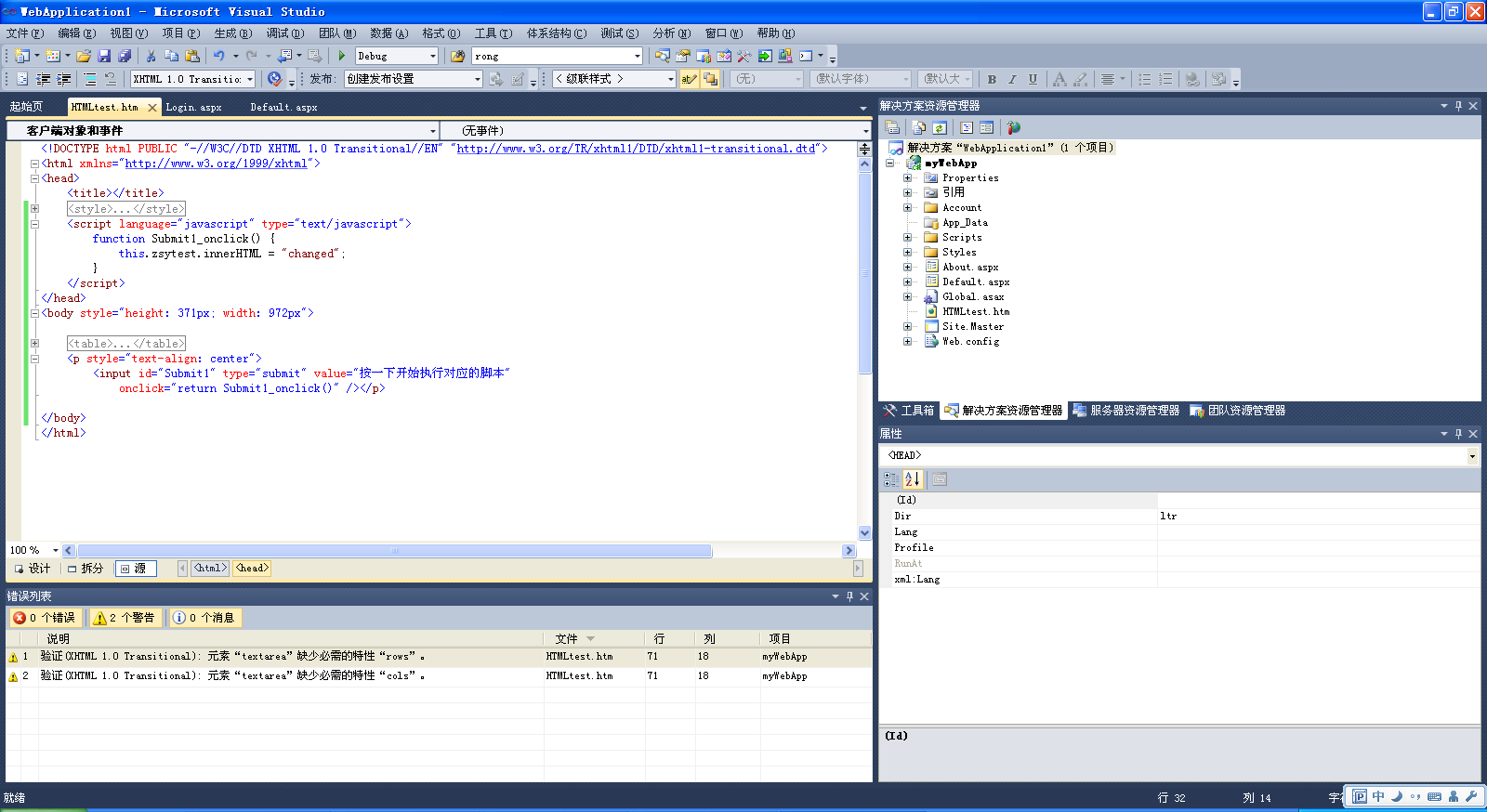
* + 1. 客户端脚本编程

客户端脚本是指在客户端执行的脚本：在html文件中，可以插入客户端脚本，浏览器可以解释执行，最常用的客户端脚本包括javascript和Vbscript。

<http://www.w3school.com.cn/js/index.asp>

应用例子：展示按下submit1按钮后的动作。

当按下submit1按钮时执行的脚本。



客户端脚本的作用：

这些脚本能够访问在浏览器安全策略范围内的客户端资源，包括用户计算机上的文件、程序、ActiveX控件、JavaApplet等，并且这些内容都需要被浏览器支持。

* 访问页面中的其他元素
* 访问本地的ActiveX控件：插入<object> </object>对象

上机实验 4：编写网页，并对网页中的各类元素进行访问。所有元素的访问参见以下的链接。<http://www.w3school.com.cn/htmldom/index.asp>，下面是对象参考手册的索引。其中表5-1是Browser对象，代表浏览器。表5-2是HTML文档对象，代表网页中的所有元素。

表5-1 Browser 对象参考手册

| **对象** | **描述** |
| --- | --- |
| [Window](http://www.w3school.com.cn/htmldom/dom_obj_window.asp) | JavaScript 层级中的顶层对象，表示浏览器窗口。 |
| [Navigator](http://www.w3school.com.cn/htmldom/dom_obj_navigator.asp) | 包含客户端浏览器的信息。 |
| [Screen](http://www.w3school.com.cn/htmldom/dom_obj_screen.asp) | 包含客户端显示屏的信息。 |
| [History](http://www.w3school.com.cn/htmldom/dom_obj_history.asp) | 包含了浏览器窗口访问过的 URL。 |
| [Location](http://www.w3school.com.cn/htmldom/dom_obj_location.asp) | 包含了当前 URL 的信息。 |

表5-2 HTML DOM（文档结构模型）

| **对象** | **描述** |
| --- | --- |
| [Document](http://www.w3school.com.cn/htmldom/dom_obj_document.asp) | 代表整个 HTML 文档，可被用来访问页面中的所有元素 |
| [Anchor](http://www.w3school.com.cn/htmldom/dom_obj_anchor.asp) | 代表 <a> 元素 |
| [Area](http://www.w3school.com.cn/htmldom/dom_obj_area.asp) | 代表图像映射中的 <area> 元素 |
| [Base](http://www.w3school.com.cn/htmldom/dom_obj_base.asp) | 代表 <base> 元素 |
| [Body](http://www.w3school.com.cn/htmldom/dom_obj_body.asp) | 代表 <body> 元素 |
| [Button](http://www.w3school.com.cn/htmldom/dom_obj_pushbutton.asp) | 代表 <button> 元素 |
| [Event](http://www.w3school.com.cn/htmldom/dom_obj_event.asp) | 代表某个事件的状态 |
| [Form](http://www.w3school.com.cn/htmldom/dom_obj_form.asp) | 代表 <form> 元素 |
| [Frame](http://www.w3school.com.cn/htmldom/dom_obj_frame.asp) | 代表 <frame> 元素 |
| [Frameset](http://www.w3school.com.cn/htmldom/dom_obj_frameset.asp) | 代表 <frameset> 元素 |
| [Iframe](http://www.w3school.com.cn/htmldom/dom_obj_iframe.asp) | 代表 <iframe> 元素 |
| [Image](http://www.w3school.com.cn/htmldom/dom_obj_image.asp) | 代表 <img> 元素 |
| [Input button](http://www.w3school.com.cn/htmldom/dom_obj_button.asp) | 代表 HTML 表单中的一个按钮 |
| [Input checkbox](http://www.w3school.com.cn/htmldom/dom_obj_checkbox.asp) | 代表 HTML 表单中的复选框 |
| [Input file](http://www.w3school.com.cn/htmldom/dom_obj_fileupload.asp) | 代表 HTML 表单中的文件上传 |
| [Input hidden](http://www.w3school.com.cn/htmldom/dom_obj_hidden.asp) | 代表 HTML 表单中的隐藏域 |
| [Input password](http://www.w3school.com.cn/htmldom/dom_obj_password.asp) | 代表 HTML 表单中的密码域 |
| [Input radio](http://www.w3school.com.cn/htmldom/dom_obj_radio.asp) | 代表 HTML 表单中的单选按钮 |
| [Input reset](http://www.w3school.com.cn/htmldom/dom_obj_reset.asp) | 代表 HTML 表单中的重置按钮 |
| [Input submit](http://www.w3school.com.cn/htmldom/dom_obj_submit.asp) | 代表 HTML 表单中的确认按钮 |
| [Input text](http://www.w3school.com.cn/htmldom/dom_obj_text.asp) | 代表 HTML 表单中的文本输入域（文本框） |
| [Link](http://www.w3school.com.cn/htmldom/dom_obj_link.asp) | 代表 <link> 元素 |
| [Meta](http://www.w3school.com.cn/htmldom/dom_obj_meta.asp) | 代表 <meta> 元素 |
| [Object](http://www.w3school.com.cn/htmldom/dom_obj_object.asp) | 代表 <Object> 元素 |
| [Option](http://www.w3school.com.cn/htmldom/dom_obj_option.asp) | 代表 <option> 元素 |
| [Select](http://www.w3school.com.cn/htmldom/dom_obj_select.asp) | 代表 HTML 表单中的选择列表 |
| [Style](http://www.w3school.com.cn/htmldom/dom_obj_style.asp) | 代表单独的样式声明 |
| [Table](http://www.w3school.com.cn/htmldom/dom_obj_table.asp) | 代表 <table> 元素 |
| [TableData](http://www.w3school.com.cn/htmldom/dom_obj_tabledata.asp) | 代表 <td> 元素 |
| [TableRow](http://www.w3school.com.cn/htmldom/dom_obj_tablerow.asp) | 代表 <tr> 元素 |
| [Textarea](http://www.w3school.com.cn/htmldom/dom_obj_textarea.asp) | 代表 <textarea> 元素 |

上机实验 5：访问本地资源

插入ActiveX控件，例子参见htmltest.htm文件，插入了pdf控件，以便显示PDF文档。用户可以观察本地还有那些ActiveX控件，可供编程使用。

*JavaApplet* 对于java Applet，html4.01以后的版本将不支持，请使用<object>标签，按照Activex的方法进行调用。

* 1. ASP动态网页编程
     1. 概述

除了将网页搞的丰富一些，漂亮一些，计算机软件的目标应该是为业务服务，所以，各网页应该有机结合，形成一个完整系统，完成业务逻辑，实现数据存储和交换。因此，需要进行动态网页设计。

*区别：动态网页是与*[*静态网页*](http://baike.baidu.com/view/348763.htm)*相对应的，也就是说，网页 URL的后缀不是.htm、.html、.shtml、.xml等静态网页的常见形式，而是以.asp、.jsp、.php、.perl、.cgi等形式为后缀。这里说的动态网页，与网页上的各种动画、滚动字幕等视觉上的“动态效果”没有直接关系，动态网页也可以是纯文字内容的，也可以是包含各种动画的内容，这些只是网页具体内容的表现形式，无论网页是否具有动态效果，采用动态网站技术生成的网页都称为动态网页。*

动态网页使用语言：HTML＋ASP 或 HTML＋PHP 或 HTML＋JSP 等。

（1）PHP即Hypertext Preprocessor(超文本预处理器)，它是当今Internet上最为火热的脚本语言，其语法借鉴了C、Java、PERL等语言，但只需要很少的编程知识你就能使用PHP建立一个真正交互的Web站点。

它与HTML语言具有非常好的兼容性，使用者可以直接在脚本代码中加入HTML标签，或者在HTML标签中加入脚本代码从而更好地实现页面控制。PHP提供了标准的数据库接口，数据库连接方便，兼容性强；扩展性强；可以进行面向对象编程。

（2）ASP即Active Server Pages，它是微软开发的一种类似HTML(超文本标识语言)、Script(脚本)与CGI(公用网关接口)的结合体，它没有提供自己专门的编程语言，而是允许用户使用许多已有的脚本语言编写ASP的应用程序。ASP的程序编制比HTML更方便且更有灵活性。它是在Web服务器端运行，运行后再将运行结果以HTML格式传送至客户端的浏览器。因此ASP与一般的脚本语言相比，要安全得多。

ASP的最大好处是可以包含HTML标签，也可以直接存取数据库及使用无限扩充的ActiveX控件，因此在程序编制上要比HTML方便而且更富有灵活性。通过使用ASP的组件和对象技术，用户可以直接使用ActiveX控件，调用对象方法和属性，以简单的方式实现强大的交互功能。

但ASP技术也非完美无缺，由于它基本上是局限于微软的操作系统平台之上，主要工作环境是微软的IIS应用程序结构，又因ActiveX对象具有平台特性，所以ASP技术不能很容易地实现在跨平台Web服务器上工作。

（3）JSP即Java Server Pages，它是由Sun Microsystem公司于1999年6月推出的新技术，是基于Java Servlet以及整个Java体系的Web开发技术。

虽然以上3种新技术在制作动态网页上各有特色，如果对微软的产品情有独钟，采用ASP技术会让您得心应手；如果是Linux的追求者，运用PHP技术在目前是最明智的选择。当然，不要忽略了JSP技术，具有跨平台的特点!

* + 1. Asp关键概念

ASP 是什么？

* ASP 指 Active Server Pages （动态服务器页面）
* ASP 是一项微软公司的技术
* ASP 是在 IIS 中运行的程序

ASP 和 HTML 有何不同？

* 当浏览器请求某个 HTML 文件时，服务器会返回这个文件
* 而当浏览器请求某个 ASP 文件时，IIS 将这个请求传递至 ASP 引擎。ASP 引擎会逐行地读取这个文件，并执行文件中的脚本。最后，ASP 文件将以纯 HTML 的形式返回到浏览器。
* Asp文件是嵌入了asp代码的HTML文件
* Asp的代码用<%> <%>嵌入到HTML中。

ASP 兼容性

* 运行 IIS，需要 Windows NT 4.0 或更高的版本。
* 运行 PWS ，需要 Windows 95 或者更高的版本。
* ChiliASP 是一种在非 Windows 操作系统上运行 ASP 的技术
* InstantASP 是另一种在非 Windows 操作系统上运行 ASP 的技术

ASP 能为你做什么？

* 动态地编辑、改变或者添加页面的任何内容
* 对由用户从 HTML 表单提交的查询或者数据作出响应
* 访问数据或者数据库，并向浏览器返回结果
* 为不同的用户定制网页，提高这些页面的可用性
  + 1. 简单ASP例子

simpleAspTest.asp

<html>

<body>

<% Response.write("Hello World!")

%>

</body>

</html>

**放在<%%>中的代码就是ASP程序。**

* + 1. ASP执行过程深入理解

前面讲过，在浏览器中通过查看源代码的方式是无法看到 ASP 源代码的，你只能看到由 ASP 文件输出的结果，而那些只是纯粹的 HTML 而已。这是因为，在结果被送回浏览器前，脚本已经在服务器上执行了。下面的实验演示了该过程。

演示实验 3**：**在本机IE中访问simpleASPtest.asp

查看本机的源程序，观察ASP的效果。

选作：用Ethereal软件观察通信过程，确切了解客户服务器交互。

* + 1. ASP编程基本规则

可将本页面中不同的<% %>中的ASP程序连起来，将其作为一个完整的程序看待，而不必去管其他的HTML标签的表现，在该程序中，只要语法规则符合脚本语言的规则，改程序就是合法的。可以声明变量、定义函数、使用脚本语言的内置函数进行运算、访问数据库服务器等。

问题：页面之间的变量如何传递？

需要强调的是：ASP在服务器端执行，因此，ASP操作的资源都是服务器端的资源。（除了cookie是写在本地的临时目录中）。

* + 1. ASP对象详解

1. ASP Response 对象：用于从服务器向用户发送输出的结果

详细参考：<http://www.w3school.com.cn/asp/asp_ref_response.asp>

重点讲解:

* response.write
* response.redirect:在需要的时候从服务器重定向到目标页面。

1. ASP Request 对象用于从用户那里取得信息

详细参考：<http://www.w3school.com.cn/asp/asp_ref_request.asp>

重点讲解：

浏览器向服务器asp页面传送信息的两种方式：

* request. QueryString 集合用于取回 HTTP 查询字符串中的变量值。解释：查询字符串的写法。

例子：带有变量的超链接。sendString.asp getString.asp

* request.Form 集合用于从使用 POST 方法的表单获取表单元素的值。解释：表单变量的提交。

例子：带有表单的页面。Submitform.asp getForm.asp

1. ASP Server 对象的作用是访问有关服务器的属性和方法

参考：<http://www.w3school.com.cn/asp/asp_ref_server.asp>

* adrot=Server.CreateObject("MSWC.AdRotator")

CreatObject可以创建服务器上所有的ActiveX控件的实例。

1. ASPError 对象用于显示在 ASP 文件的脚本中发生的任何错误的详细信息

参考：<http://www.w3school.com.cn/asp/asp_ref_error.asp>

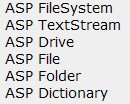
用于调试和出错处理。

1. FileSystemObject 对象用于访问服务器上的文件系统。

* fs=Server.CreateObject("Scripting.FileSystemObject")

然后可以利用fs的方法提供文件系统的访问。

能够访问：



1. Application、Session和Cookies对象

通过前面讲解，我们知道客户端页面如何向服务器端页面传送信息，并且知道同一页面内的变量可以访问。问题是，不同页面中的变量如何相互访问？

Asp提供了三种方式：

* Application---应用程序对象，可以存储所有用户共用的变量，一旦网站启动，则建立全局应用程序对象，直到网站关闭。例如：

application("vartime")=""

application("users")=1

还可以锁定Application

Application.Lock

Application.Unlock

* session---会话程序对象，一旦某个用户连接到网站，则建立会话对象，存储该用户的信息，用户关闭网站，则会话结束，如果超出规定的时间没有活动，则会话也结束。

<%

Session("username")="Donald Duck"

Session("age")=50%>

用Session.content对象访问所有的session变量。

Session.Abandon，终止会话。

%>

* cookie---cookie 是一种服务器留在用户电脑中的小文件。参考<http://www.w3school.com.cn/asp/asp_cookies.asp>

Response.cookie: 可将信息保存在用户计算机上以供其他页面读取。

<% Response.Cookies("firstname")="Alex" %>

"Request.Cookies" 命令用于取回 cookie 的值。

fname=Request.Cookies("firstname")

上机实验 6**：基本ASP对象编程练习**

编写一个登录界面，收集用户名、密码信息，然后进行提交。

验证界面收集用户名和密码，（进行验证，正确后）自动转向主页面，在主页面中显示欢迎该用户的信息，以及当前用户数量的信息，用两种方法实现用户名信息的传递。（）内的步骤暂时省略。

* + 1. 完整的ASP应用程序
       1. Global.asa 文件

Global.asa 文件是一个可选的文件，它可包含可被 ASP 应用程序中每个页面访问的对象、变量以及方法的声明。所有合法的浏览器脚本都能在 Global.asa 中使用。

参见：<http://www.w3school.com.cn/asp/asp_globalasa.asp>

Global.asa 文件可包含下列内容：

* Application 事件
* Session 事件
* <object> 声明
* TypeLibrary 声明
* #include 指令

通常将共用的信息放到该文件中，例如数据库连接串等。

演示例子：webAPP中的Gloable.asa文件。

* + - 1. include文件

通过使用 #include 指令，我们可以在服务器执行 ASP 文件之前，把另一个ASP文件插入这个文件中。#include 命令用于在多个页面上创建需要重复使用的函数、页眉、页脚或者其他元素等。实现重用。

通常将Include文件放在一个Include文件夹中，并将被包含的文件命名为.inc

例子：submitForm.asp

* + - 1. 样式文件（CSS文件）

<http://www.w3school.com.cn/css/css_intro.asp>，CSS 指层叠样式表 (Cascading Style Sheets)

* 样式定义如何显示 HTML 元素
* 样式通常存储在样式表中
* 把样式添加到 HTML 4.0 中，是为了解决内容与表现分离的问题
* 外部样式表可以极大提高工作效率
* 外部样式表通常存储在 CSS 文件中
* 多个样式定义可层叠为一个

讲解CSS 基础语法<http://www.w3school.com.cn/css/css_syntax.asp>

读例子，理解css的编写。<http://www.w3school.com.cn/example/csse_examples.asp>

样式编辑器。关键在于样式的设计。

* + 1. ASP访问数据库（ADO）

从 ASP 页面访问数据库的通常途径是：

* 创建至数据库的 ADO 连接（ADO connection）
* 打开数据库连接
* 创建 ADO 记录集（ADO recordset）
* 打开记录集（recordset）
* 从数据集中提取你所需要的数据
* 关闭数据集
* 关闭连接

演示：一个简单的ASP访问程序：simpleADO.asp

演示：一个复杂的ASP访问数据库程序。Ado.asp

数据的增删改

添加记录：<http://www.w3school.com.cn/ado/ado_add.asp>

修改记录：<http://www.w3school.com.cn/ado/ado_update.asp>

删除记录：<http://www.w3school.com.cn/ado/ado_delete.asp>

上机实验 7：将上面的用户登录实验中的用户信息通过数据库验证。即实现（）中的内容。

* 1. B/S程序开发流程

与通用程序一样，B/S程序的开发也是按照软件工程的流程进行，需要经历需求分析、概要设计、详细设计、实现、测试、维护各个阶段。

由于B/S架构程序中页面的展示性用户较为关注，因此，在需求分析、设计方面有其特点。下面的链接的前半部分，可以启发一下思路。

<http://www.tudou.com/programs/view/RN8zRZvF9HI/>



页面设计分为程序设计和美工设计。在公司里，一般都有专门的美工对页面进行设计，当然，这些美工也不是纯粹的美工，需要对网页的功能和机理有一定的了解，提供样板，然后，开发人员根据样板对页面进行设计。下面的链接给出了一个利用模版进行设计的例子。

<http://www.tudou.com/programs/view/KTAj_Ts1aPk/>

* 1. 基于CSS的页面布局
     1. 布局概述

前端设计包括脚本设计和页面布局等，根据应用需求的不同，对于两层架构的设计主要指前端设计和后端数据库设计，前端设计主要是进行页面设计和页面之间的接口设计。三层架构包括前端设计、服务器端程序设计、数据库设计。本部分主要介绍页面布局。

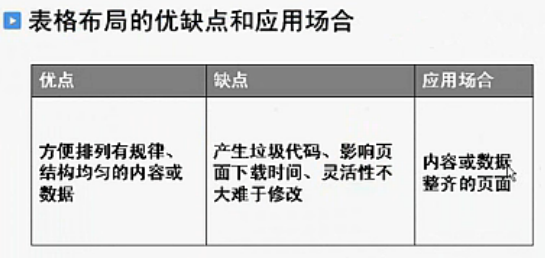
页面布局一般分为三种：框架布局、表格布局和基于Div块和CSS的布局。

（1）框架布局



框架使用框架结构标签<frameset>元素来实现，<frameset>定义如何将窗口分割为框架，每个 frameset 定义了一系列行或列参考：<http://www.w3school.com.cn/html/html_frames.asp>

（2）表格布局



表格布局使用<table>元素实现。具体参考<http://www.w3school.com.cn/html/html_tables.asp>。

（3）DIV块布局模型



CSS 定位 (Positioning) 属性允许你对元素进行定位，从而实现布局。在布局中，将元素都放到块中。div、h1 或 p 元素常常被称为块级元素。这意味着这些元素显示为一块内容，即“块框”。与之相反，span 和 strong 等元素称为“行内元素”，这是因为它们的内容显示在行中，即“行内框”。

定位的基本思想很简单，它允许你定义元素框相对于其正常位置应该出现的位置，或者相对于父元素、另一个元素甚至浏览器窗口本身的位置。

您可以使用 display 属性改变生成的框的类型。这意味着，通过将 display 属性设置为 block，可以让行内元素（比如 <a> 元素）表现得像块级元素一样。还可以通过把 display 设置为 none，让生成的元素根本没有框。这样的话，该框及其所有内容就不再显示，不占用文档中的空间。

但是在一种情况下，即使没有进行显式定义，也会创建块级元素。这种情况发生在把一些文本添加到一个块级元素（比如 div）的开头。即使没有把这些文本定义为段落，它也会被当作段落对待：

<div>

some text

<p>Some more text.</p>

</div>

在这种情况下，这个框称为无名块框，因为它不与专门定义的元素相关联。

块级元素的文本行也会发生类似的情况。假设有一个包含三行文本的段落。每行文本形成一个无名框。无法直接对无名块或行框应用样式，因为没有可以应用样式的地方（注意，行框和行内框是两个概念）。但是，这有助于理解在屏幕上看到的所有东西都形成某种框。

目前，主流的、复杂的业务都是用这类布局方式，下面对CSS布局的基本概念和方法进行介绍。

* + 1. CSS框模型 (Box Model)
       1. 模型

CSS 框模型 (Box Model) 规定了元素框处理元素内容、[内边距](http://www.w3school.com.cn/css/css_padding.asp)、[边框](http://www.w3school.com.cn/css/css_border.asp) 和 [外边距](http://www.w3school.com.cn/css/css_margin.asp) 的方式。

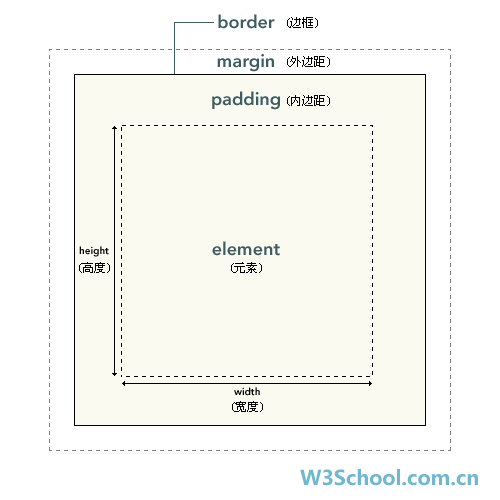
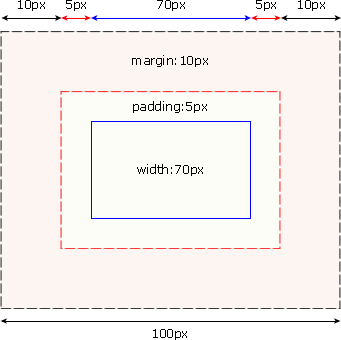


图5-1 CSS 框模型

元素框的最内部分是实际的内容，直接包围内容的是内边距。内边距呈现了元素的背景。内边距的边缘是边框。边框以外是外边距，外边距默认是透明的，因此不会遮挡其后的任何元素。

*提示：1）背景应用于由内容和内边距、边框组成的区域；2）内边距、边框和外边距都是可选的，默认值是零；3）在 CSS 中，width 和 height 指的是内容区域的宽度和高度。增加内边距、边框和外边距不会影响内容区域的尺寸，但是会增加元素框的总尺寸。*

假设框的每个边上有 10 个像素的外边距和 5 个像素的内边距。如果希望这个元素框达到 100 个像素，就需要将内容的宽度设置为 70 像素，请看下图：



#box {

width: 70px;

margin: 10px;

padding: 5px;

}

提示：内边距、边框和外边距可以应用于一个元素的所有边，也可以应用于单独的边。外边距可以是负值，而且在很多情况下都要使用负值的外边距。

* + - 1. 内边距

元素的内边距在边框和内容区之间。控制该区域最简单的属性是 padding 属性。

CSS padding 属性定义元素边框与元素内容之间的空白区域。

CSS padding 属性定义元素的内边距。padding 属性接受长度值或百分比值，但不允许使用负值。百分数值是相对于其父元素的 width 计算的，如果父元素的 width 改变，它们也会改变。

下面这条规则把段落的内边距设置为父元素 width 的 10%：

p {padding: 10%;}

例如：如果一个段落的父元素是 div 元素，那么它的内边距要根据 div 的 width 计算。

<div style="width: 200px;">

<p>This paragragh is contained within a DIV that has a width of 200 pixels.</p>

</div>

注意：上下内边距与左右内边距一致；即上下内边距的百分数会相对于父元素宽度设置，而不是相对于高度。

* + - 1. 边框

元素的边框 (border) 是围绕元素内容和内边距的一条或多条线。

CSS border 属性允许你规定元素边框的样式、宽度和颜色。

在 HTML 中，可以使用表格来创建文本周围的边框，但是通过使用 CSS 边框属性，我们可以创建出更为效果出色的边框，并且可以应用于任何元素。

每个边框有 3 个方面：宽度、样式，以及颜色。

h1 {

border-style: solid;

border-color: black;

border-right-color: red;

}

* + - 1. 外边距

围绕在元素边框的空白区域是外边距。设置外边距会在元素外创建额外的“空白”。

设置外边距的最简单的方法就是使用 margin 属性，这个属性接受任何长度单位、百分数值甚至负值，包括：像素、英寸、毫米或 em。

margin 可以设置为 auto。更常见的做法是为外边距设置长度值。下面的声明在 h1 元素的各个边上设置了 1/4 英寸宽的空白：

h1 {margin : 0.25in;}

注意：当两个垂直外边距相遇时，它们将形成一个外边距。合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

* + 1. CSS 布局机制和实现方式

CSS 有三种基本的布局机制：普通流、浮动和定位。

* + - 1. 普通流

**普通流：**顾名思义，就是根据块级元素的标签在HTML里的顺序，像水流一样，从上至下。当然对于行内元素而言，还是在一行中水平排列的。

*复习概念：HTML 块元素，大多数 HTML 元素被定义为块级元素或内联元素。 “块级元素”译为 block level element，“内联元素”译为 inline element。块级元素在浏览器显示时，通常会以新行来开始（和结束）。*

*例子：<h1>, <p>, <ul>, <table>*

*HTML 内联元素（行内元素）：内联元素在显示时通常不会以新行开始。*

*例子：<b>, <td>, <a>, <img>*

行内元素可以在水平（内间距、边框。外边距）方向上修改它们水平尺寸，但是在垂直方向上对行内元素的高度是毫无影响的，还有就是直接定义行内元素的 width/height也是毫无影响。对与行内元素来说，它的高度仅仅容纳下它所包含的内容即可。但是通过设置行内元素的行高line- height，就可以增加这个行内元素的高度。( 当然还有就是，将行内元素的style属性里设置一个display:block，即可将其转化成块级元素。)

* + - 1. 定位

（1）概述

**相对定位(relative):** 是一种相对于起点的移动，根据top和left值做出改变。但是无论是否移动，元素仍然占据原来的空间，所以移动的元素会导致覆盖的情况。它仍然是被看作普通文档流的一部分。

**绝对定位(absolute):** 这是一种非常有用，也经常被人滥用的css技术。绝对定位会使元素脱离普通文档流，不占据文档流的空间。其它元素的布局会忽视绝对定位。它是相对于基于它的上一个已经定位的祖先元素进行偏移。若没有已经定位的祖先元素，通常情况下就会根据HTML元素进行偏移。进行绝对定位后，会有显示的层级z- index，数值越高，层级越高。

除非专门指定，否则所有框都在普通流中定位。也就是说，普通流中的元素的位置由元素在 (X)HTML 中的位置决定。块级框从上到下一个接一个地排列，框之间的垂直距离是由框的垂直外边距计算出来。行内框在一行中水平布置。可以使用水平内边距、边框和外边距调整它们的间距。但是，垂直内边距、边框和外边距不影响行内框的高度。由一行形成的水平框称为行框（Line Box），行框的高度总是足以容纳它包含的所有行内框。不过，设置行高可以增加这个框的高度。

（2）实现方式

相对定位和绝对定位通过CSS position 属性实现，除了这两种以外， [position 属性](http://www.w3school.com.cn/cssref/pr_class_position.asp)总共提供了 4 种不同类型的定位，这会影响元素框生成的方式。通过 float 属性实现元素的浮动

position 属性值的含义

**Static：**元素框正常生成。块级元素生成一个矩形框，作为文档流的一部分，行内元素则会创建一个或多个行框，置于其父元素中。

**Relative：**元素框偏移某个距离。元素仍保持其未定位前的形状，它原本所占的空间仍保留。

如果将 top 设置为 20px，那么框将在原位置顶部下面 20 像素的地方。如果 left 设置为 30 像素，那么会在元素左边创建 30 像素的空间，也就是将元素向右移动。

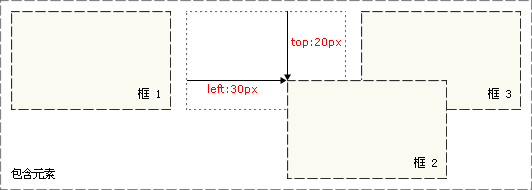
#box\_relative {

position: relative;

left: 30px;

top: 20px;}

效果如下图所示：



*注意，在使用相对定位时，无论是否进行移动，元素仍然占据原来的空间。因此，移动元素会导致它覆盖其它框。*

**Absolute：**元素框从文档流完全删除，并相对于其包含块定位。包含块可能是文档中的另一个元素或者是初始包含块。元素原先在正常文档流中所占的空间会关闭，就好像元素原来不存在一样。元素定位后生成一个块级框，而不论原来它在正常流中生成何种类型的框。

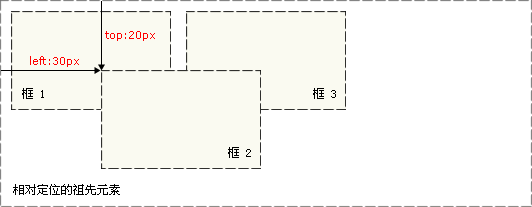
#box\_absolute {

position: absolute;

left: 30px;

top: 20px;}

如下图所示：



**Fixed：**元素框的表现类似于absolute，不过其包含块是视窗本身。

* + - 1. 浮动

浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。由于浮动框不在文档的普通流中，所以文档的普通流中的块框表现得就像浮动框不存在一样。

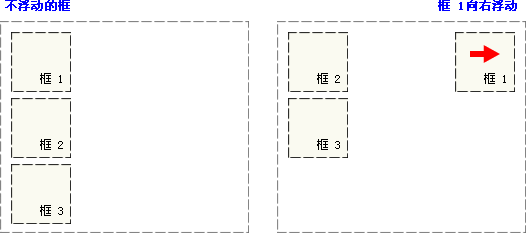
浮动利用Float属性实现，CSS float 属性语法：

{float:right|left|none|inherit; }

参考：http://www.w3school.com.cn/cssref/pr\_class\_float.asp

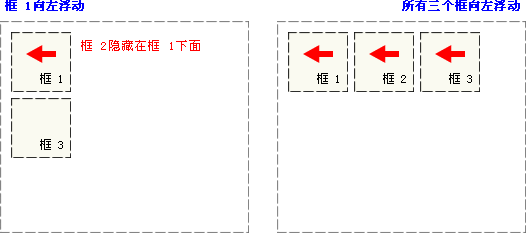
浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。由于浮动框不在文档的普通流中，所以文档的普通流中的块框表现得就像浮动框不存在一样。

请看下图，当把框 1 向右浮动时，它脱离文档流并且向右移动，直到它的右边缘碰到包含框的右边缘：

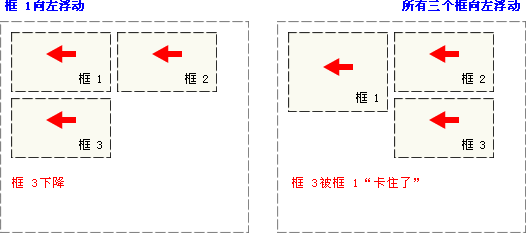


再请看下图，当框 1 向左浮动时，它脱离文档流并且向左移动，直到它的左边缘碰到包含框的左边缘。因为它不再处于文档流中，所以它不占据空间，实际上覆盖住了框 2，使框 2 从视图中消失。

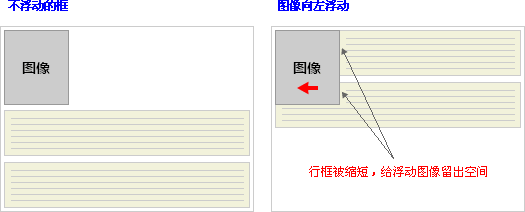
如果把所有三个框都向左移动，那么框 1 向左浮动直到碰到包含框，另外两个框向左浮动直到碰到前一个浮动框。



如下图所示，如果包含框太窄，无法容纳水平排列的三个浮动元素，那么其它浮动块向下移动，直到有足够的空间。如果浮动元素的高度不同，那么当它们向下移动时可能被其它浮动元素“卡住”。



行框和清理：浮动框旁边的行框被缩短，从而给浮动框留出空间，行框围绕浮动框。因此，创建浮动框可以使文本围绕图像：



要想阻止行框围绕浮动框，需要对该框应用 [。属性](http://www.w3school.com.cn/cssref/pr_class_clear.asp)。clear 属性的值可以是 left、right、both 或 none，它表示框的哪些边不应该挨着浮动框。

为了实现这种效果，在被清理的元素的上外边距上添加足够的空间，使元素的顶边缘垂直下降到浮动框下面：



这是一个有用的工具，它让周围的元素为浮动元素留出空间。

让我们更详细地看看浮动和清理。假设希望让一个图片浮动到文本块的左边，并且希望这幅图片和文本包含在另一个具有背景颜色和边框的元素中。您可能编写下面的代码：

.news {

background-color: gray;

border: solid 1px black;

}

.news img {

float: left;

}

.news p {

float: right;

}

<div class="news">

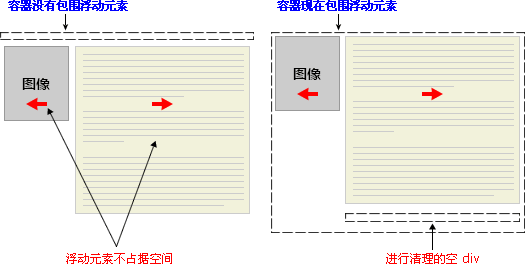
<img src="news-pic.jpg" />

<p>some text</p>

</div>

这种情况下，出现了一个问题。因为浮动元素脱离了文档流，所以包围图片和文本的 div 不占据空间。

如何让包围元素在视觉上包围浮动元素呢？需要在这个元素中的某个地方应用 clear：



不幸的是出现了一个新的问题，由于没有现有的元素可以应用清理，所以我们只能添加一个空元素并且清理它。

.news {

background-color: gray;

border: solid 1px black;

}

.news img {

float: left;

}

.news p {

float: right;

}

.clear {

clear: both;

}

<div class="news">

<img src="news-pic.jpg" />

<p>some text</p>

<div class="clear"></div>

</div>

这样可以实现我们希望的效果，但是需要添加多余的代码。常常有元素可以应用 clear，但是有时候不得不为了进行布局而添加无意义的标记。

不过我们还有另一种办法，那就是对容器 div 进行浮动：

.news {

background-color: gray;

border: solid 1px black;

float: left;

}

.news img {

float: left;

}

.news p {

float: right;

}

<div class="news">

<img src="news-pic.jpg" />

<p>some text</p>

</div>

这样会得到我们希望的效果。不幸的是，下一个元素会受到这个浮动元素的影响。为了解决这个问题，有些人选择对布局中的所有东西进行浮动，然后使用适当的有意义的元素（常常是站点的页脚）对这些浮动进行清理。这有助于减少或消除不必要的标记。

事实上，W3School 站点上的所有页面都采用了这种技术，如果您打开我们使用 CSS 文件，您会看到我们对页脚的 div 进行了清理，而页脚上面的三个 div 都向左浮动。

1. XML语言及Ajax
   1. XML基础

XML简介：<http://www.w3school.com.cn/xml/xml_intro.asp>

XML 的用途：<http://www.w3school.com.cn/xml/xml_usedfor.asp>

XML语法：<http://www.w3school.com.cn/xml/xml_syntax.asp>

XML元素：<http://www.w3school.com.cn/xml/xml_elements.asp>

XML属性：<http://www.w3school.com.cn/xml/xml_attributes.asp>

XML验证：<http://www.w3school.com.cn/xml/xml_dtd.asp>

其他深入学习请参考: <http://www.w3school.com.cn/xml/>

* 1. Ajax开发技术

AJAX 指异步 JavaScript 及 XML（Asynchronous JavaScript And XML）。

AJAX 是一种在 2005 年由 Google 推广开来的编程模式。

AJAX 不是一种新的编程语言，而是一种使用现有标准的新方法。

AJAX 实例：<http://www.w3school.com.cn/ajax/ajax_example.asp>

AJAX创建XMLHttpRequest 对象：[http://www.w3school.com.cn/ajax/ajax\_ xmlhttprequest\_create.asp](http://www.w3school.com.cn/ajax/ajax_%20xmlhttprequest_create.asp)

AJAX - 向服务器发送请求

AJAX - 服务器响应：responseText，responseXML，使用XMLdom对象对XML文档进行解析。

1. JSP Web开发基本架构
   1. Java静态页面技术支持

Java Applet就是用Java语言编写的一些小应用程序，它们可以直接嵌入到网页中，并能够产生特殊的效果。包含Applet的网页被称为Java-Powered页，可以称其为Java支持的网页。

当用户访问这样的网页时,Applet被下载到用户计算机上执行，但前提是用户使用的是支持Java的网络浏览器。由于Applet是在用户计算机上执行的，下载以后，它的执行速度是不受网络宽带或者MODEM存取速度的限制，用户可以更好的欣赏网页上Applet产生的多媒体效果。

在Java Applet中，可以实现图形绘制、字体和颜色控制、动画和声音的插入、人机交互及网络交流等功能.Apple还提供各种名为抽象窗口工具箱 (Abstract Window Toolkit,AWT)的窗口环境开发工具,但AWT的设计是存在缺陷的,取而代之的Swing技术为Java GUI组建的轻量组建.它们利用用户计算机的GUI元素,可以建立标准的图形用户界面,如窗口、按钮、滚动条等。目前，在网络上有非常多的Applet范例来生动地展现这些功能。

* + 1. 创建Applet

开发环境NetBeans

Before you create an applet in the IDE, you must determine the type of project from which you want to run the applet. You can create the applet in one of the following two places:

* Within a Java Application project with Java WebStart enabled and with an applet descriptor specified in the application's JNLP file.
* Within a Java Class Library project that is a library for a Web Application project.

*参考NetBeans帮助。Creat an applet.*

下面的例子采用第二种方法创建Applet，并将其生成Jar包，发布到java web项目中。

步骤：

（1）建立JavaApplication项目，添加myJapplet.java文件（Japplet模版生成），将其放到test包下。



（2）编辑文件；

（3）生成jar包，在dist目录下；

（4）将该文件发布到webAppJava项目中，如下图所示。



* + 1. 使用Applet

（1）编辑文件Index.jsp，插入Applet

<applet name="myApp" code="test.myJApplet" archive="JavaApplet.jar" width=600 height=400 >

</applet>

（2）调用Applet中的方法

<script language="JavaScript">

myApp.paint(); //调用Applet并执行其中的方法。

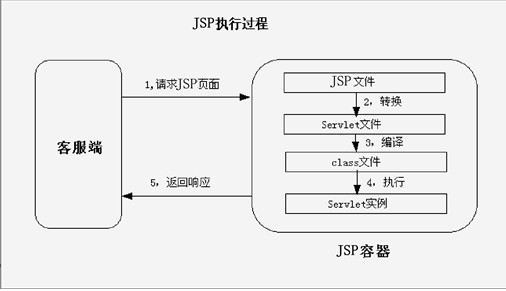
</script>

演示实验 4**：**演示NetBeans中的上述程序，在JavaApplet中看test目录下的myJapplet，在webAppJava中看Index.jsp中的程序，然后执行观看效果。

* 1. JSP程序基本结构

JavaServer PagesTM technology, abbreviated as JSP, provides a simplified, fast way to create web pages and applications that display dynamically generated content. Such content includes HTML, DHTML, XHTML, and XML. In essence, a JSP page describes how to process a request to create a response.

A JSP page is a server-side, text-based J2EETM component. Such a component executes in a container installed on a web server or on a web-enabled application server. The JSP container (www服务)delivers requests from a client to a JSP page and responses from the JSP page to the client. When compiling a JSP page, the container dynamically translates the JSP source into a JavaTM class that implements the javax.servlet.Servlet interface.



A JSP page looks like a standard HTML or XML page, with additional directive elements, scripting elements, and action elements that the JSP engine processes and strips out. Typically, JSP elements create text that is inserted into the results page.

There are three types of JSP pages:

**Standard JSP pages.** Standard JSP pages are written using the [standard JSP syntax](file:///E:\zsy\教学\Internet\about_jsp_syntax.html). Standard JSP pages typically use the .jsp extension. The JSP code is in XHTML format, and follows standard JSP conventions for the XML representation of a JSP page.

**JSP documents.** JSP documents are written using the JSP document syntax, which is well formed XML. JSP documents typically have the .jspx extension. Two examples of the advantages that JSP documents have over standard JSP pages are:

You can edit, verify, and view them with XML capable tools, such as the IDE's Source Editor.

You can transform them using XSLT tools, such as the IDE's XSL Transformation command.

**JSP segments.** A segment is a file that contains a fragment of JSP text for inclusion by standard JSP pages and JSP documents. JSP segments typically use the .jspf extension.

For additional information, see the *JavaServer Pages Specification* available at

与ASP类似，JSP将服务器端的脚本嵌入到HTML文件中，在请求时编译成java执行代码，由Java虚拟机执行，生成客户请求HTML文本，返回给客户。

JSP中服务器端代码嵌入到<% %>中，服务器代码使用JavaScript脚本语言，这段代码称为Scriptlet.

例子：

<%@ page contentType="text/html; charset=gb2312" %>

<%!

String hello(){

return "欢迎进入**JSP**，^\_^";}

%>

<html>

<HEAD>

<meta http-equiv="Content-Type" content="text/html; charset=gb2312">

<title>在**JSP**中定义函数</title>

</HEAD>

<body>

<%=hello()%>

</body>

</html>

* 1. JSP指令和对象
     1. 编译指令

JSP的三个编译指令：

（1）Page:该指令是针对当前页面的指令 。

语法：<%@ page [Attribute="Value"] %>。其属性包括以下：

language:声明当前jsp页面使用的脚本语言的种类,通常是java,默认是java,无需设置。

extends:指定jsp编译产生的类继承的父类,或者实现的接口；

import:用于导入包,默认已经导入的包有 java.lang.\*,javax.servlet.\*,javax.servlet.jsp.\*,javax.servlet.http.\*;

session:设定这个jsp页面是否需要HttpSession；

buffer:指定输出缓冲区的大小；

autoflush:当缓冲区即将溢出时,是否自动强制输出缓冲区的内容；

info:设置该JSp程序的信息,采用Servlet.getServletInfo()获取该值；

errorPage:指定错误处理页；

isErrorPage:设置本JSp是否为错误处理程序；

ContentType:用于设定生成网页的文件格式和编码格式。

（2）include:用于指定包含另一个页面

语法：<%@include file="path"%>

（3）taglib:用于定义和访问自定义标签。

语法：<%@ taglib uri="tagLibraryURI" prefix="tagPrefix" %>

uri (不是url) 描述引用的标签库的位置 如: /WEB-INF/struts-bean.tld

prefix 指该页面使用此标签库的前缀区分同名标签，例如：将prefix设为myPrefix时，在标签库中有一myTag标签，可在页面中<myPrefix:myTag>。下面这些前缀已经保留：jsp:,jspx:,javax:,servlet:,sun:,sunw:

* + 1. 运行时指令

JSP的7个运行时指令，用于在运行时执行相应的动作。

jsp:forward 执行页面转向,将请求的处理交给下一个页面。<jsp:forward page="path<%expression%>">

jsp:param 传递参数。<jsp:param name="" value="" />。<jsp:forward>常和<jsp:param>指令一起。jsp:param必须与其他支持参数的标签一起使用；

jsp:include 用于动态引入一个jsp页面。<jsp:include page="<url|expressions>" flush='true'/> 也可以在标签内向引用页面传递param；

jsp:plugin 用于下载javaBean 或Applet到客户端执行；

jsp:useBean 创建一个javaBean实例。<jsp:useBean id="name" class="classname" scope="page|request|session|application" />，scope属性：page:该javaBean实例仅在本页面有效；request: 仅本次请求有效；session:在本次session有效；application:在本应用内一直有效；

jsp:setProperty 设置javaBean实例的属性值。<jsp:setProperty name="BeanName" property="propertyName" value="" />；

jsp:getProperty 取得JavaBean实例的属性值 <jsp:getProperty name="beanName" property="propertyName">

* + 1. JSP内置组件

**JSP**共有以下9种基本内置组件（可与ASP的6种内部组件相对应）：

|  |
| --- |
| **request** 用户端请求，此请求会包含来自GET/POST请求的参数  **response** 网页传回用户端的回应  **pageContext** 网页的属性是在这里管理  **session** 与请求有关的会话期  **application** **Servlet**正在执行的内容  **out** 用来传送回应的输出  **config** **Servlet**的构架部件  **page** **JSP**网页本身  **exception** 针对错误网页，未捕捉的例外 |

例如：

<% String name=request.getParameter("name");

out.println(name); %>

又如：

<%

name = request.getParameter("thename");

session.putValue("thename", name); //将name的值保存在session变量thename中。

%>

其他使用方法参考相关手册。

例子参见项目：JSPExamples的执行效果

* 1. JSP与Java Beans

**JavaBeans** 就是一个可以复用软件模型，**JavaBeans** 在某个容器中运行，提供具体的操作性能。

在**JSP** 程序中能够引用JavaBeans，以实现软件复用，所用的**JavaBean**s一般以不可见的组件为主，可见的**JavaBeans**一般用于编写Applet 程序或者**Java** 应用程序。



**JavaBean**s设计中，按照属性的不同作用又细分为四类：**Simple, Index, Bound**与**Constrained**属性。详细参考相关资料。**Java** 应用程序在运行时，最终用户也可以通过**JavaBeans**所提供的属性存取方法（**setXXX** 方法和**getXXX）**方法修改**JavaBeans** 组件的属性。

* + 1. Java Bean的引用

要在**JSP**使用"**Bean**"，首先必须在**jsp**设置要引用的**Bean**，生成**Bean**的一个实例。可以用“**<jsp:useBean>**”标记来完成：

**<jsp:useBean id="Name" class="com.testbean" scope="application" />。**

"**<jsp:useBean>**"标记是通过“id”属性来识鉴别**Bean**。当指定了"id"属性后，还须告诉网页面从何处查找**Bean**，或者它的**Java**类别名是什么。最后一个必需的元素是“scope”属性。有了“**scope**”属性的帮助，你就能告诉Bean为单一页面（默认情况）[scope="page"]、为一个被请求的网页[**scope="request"**]、请求为会话[**scope="session"**]，或者为整个应用程序[**scope="application"**]保持留信息。有了会话范围，你就能非常容易地在**JSP**网页面上维护网站上的项目，如购物车项目等。

实例化一个**JavaBean**，就可以访问它的属性来定制它。要获得属性值，请用“**<jsp:getProperty>**”标记。使用这个标记能指定将要用到的Bean名称(从useBean的"id"字段得到)以及你想得到其值的属性。实际的值被放在输出中：

**<jsp:getProperty id="Name" property="name" />。**

要改变**JavaBean**属性，你必须使用“<jsp:setProperty>”标记。对这个标记，需要再次识别Bean和其属性，以修改并额外提供新值。如果命名正确，这些值可以从一个已提交的表中直接获得：参数获得：

<jsp:setProperty id="Name" property="\*" />;

可以从一个参数获得，须直接命名属性和参数：

<jsp:setProperty id="Name" property="serialNumber" value="string" />

或者直接用一个名字称和值来设置：

<jsp:setProperty id="Name" property="serialNumber" value= <%= expression %> />.

* + 1. Java Beans实验

演示实验 5：javabean及调用

**参见WebAppJava中beans/NewBean.java，调用方法在index.jsp中**

package beans;

import java.beans.\*;

import java.io.Serializable;

import java.util.Date;

import java.text.\*;

/\*\*

\*

\* @author Administrator

\*/

public class NewBean implements Serializable {

public static final String PROP\_SAMPLE\_PROPERTY = "sampleProperty";

private String myP;

private PropertyChangeSupport propertySupport;

Date d=new Date();

public String now(String s)

{

SimpleDateFormat formatter = new SimpleDateFormat(s);

return formatter.format(d);

}

public NewBean() {

propertySupport = new PropertyChangeSupport(this);

}

public String getSampleProperty() {

return myP;

}

public void setSampleProperty(String value) {

String oldValue = myP;

myP = value;

propertySupport.firePropertyChange(PROP\_SAMPLE\_PROPERTY, oldValue, myP);

}

public void addPropertyChangeListener(PropertyChangeListener listener) {

propertySupport.addPropertyChangeListener(listener);

}

public void removePropertyChangeListener(PropertyChangeListener listener) {

propertySupport.removePropertyChangeListener(listener);

}

}

测试页面的代码为：date.jsp

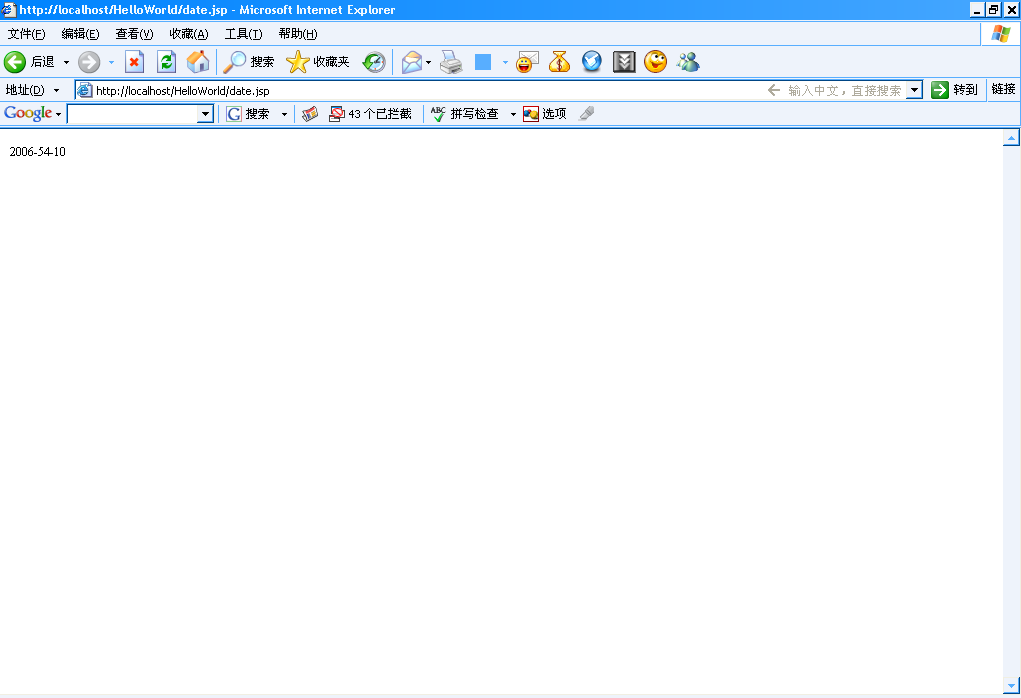
<jsp:useBean scope="page" id="dt" class="beans.Newbean"/>

<%

out.print(dt.now("yyyy-mm-dd"));

%>

输出结果：



* 1. Java Servlet

Servlets是Java专注于CGI开发的一种技术。运行在Server端，并产生动态的结果。为什么要使用Servlets来代替传统的CGI程序呢？

效率：使用传统的CGI程序，每当收到一个HTTP请求的时候，系统就要启动一个新的进程来处理这个请求，这样会导致系统性能的降低。而使用Servlets，Java VMs一直在运行，当接到一个请求之后Java VMs就创建一个Java线程马上进行处理，如此要比每次都启动一个新的系统进程效率要高的多。

（1）Servlets特点：

功能强大：Servlets使你可以使用传统CGI不能提供的许多强大功能。你可以使用Java的API 去完成任何传统CGI认为困难或不可能的事情。Servlets可以轻松的实现数据共享和信息维护，跟踪session和其他功能。

安全：Servlets运行在Servlets引擎的限制范围之内，就象可以在Web浏览器中运行Applets一样，这样有助于保护Servlets不受威胁。

成本：由于Servlets可以运行在多个Web服务器上，这样你就可以使用免费或价格便宜的服务器，并让他支持Servlets，如此可以大大减少成本开支。

灵活性：由于Servlets是在Java平台上运行的，所以由于Java的跨平台性，Servlets也可以从一个平台轻易地转移到另一个操作系统平台上，从而大大提高了灵活性。

一个Servlets实际上就是一个Java类，需要运行在Java的虚拟机上，使用Servlets引擎。当某个Servlets被请求的时候，Servlets引擎调用该Servlets并一直运行到这个被调用的Servlets运行完毕或Servlets引擎被关闭。

（2）JavaServlets的其他属性

由于JAVA是面向对象的语言,因此JAVA平台已经将HTTP转换成对象的形式.这将会使JAVA 程序员关注于应用本身而不是HTTP。

HTTP提供了一个标准的机制来扩展服务器的功能,我们称之为CGI.服务器将请求发送到CGI程序,CGI程序则返回一个响应.同样的任何JAVA服务器则会接收请求,然后转发到Servlet。

Servelt 是javax.servlet.http.HttpServlet的子类,每个Servlet必须完成四个方法：

\* public void init(ServletConfig config)

\* public void doGet(HttpServletRequest request, HttpServletResponse response)

\* public void doPost(HttpServletRequest request, HttpServletResponse response)

\* public void destroy()

（一）Servlets 和多线程

为了提高性能,Servlet设计程多线程.每个Servlet仅创建一个实例,每一个请求都传递到同一个对象.这将有利于Servlet容器充分的利用资源.因此doGet,doPos在编程时必须保证他是线程安全的。

%CATALINA\_HOME%\lib;%CATALINA\_HOME%\lib\servlet-api.jar;%CATALINA\_HOME%\lib\jsp-api.jar

（二）Servlet Context

ServletContext(javax.servlet.servletContext)定义了Web 应用中Servlet的视图.在Servlet中通过getServletConfig()可以访问得到,在JSP中则通过隐式对象application得到.servletContext提供了几个对于创建Struts应用来说非常有用的几个方法:

1、访问WEB应用资源：Servlet通过getResource(),getResourceAsStream()可以访问WEB应用中的静态资源文件。

2、Servlet Context属性：Servlet上下文可以存储JAVA对象到属性中.这些属性对整个WEB应用都可见。

（三）、Servlet请求

Servlet请求就是javax.servlet.http.HttpServletRequest,通过它可以访问HTTP请求的所有信息:

1、Cookies：通过getCookies()可以得到当前请求的所有cookie

2、HTTP头：HTTP请求的头可以通过对应的名字来访问.你当然可以通过枚举来列出所有的头。

3、参数：你可以通过参数来访问HTTP请求的URL的参数或表单中的内容.

4、请求特性：HTTP请求表单的提交方式(GET/POST),用的是什么协议(HTTP/HTTPS)

5、请求URI信息：通过getRequestURI()可以得到最初的请求URI,除此之外,我们还可以得到contextPath,ServletPath,pathInfo。

6、用户信息：如果你正使用容器来进行安全管理,你可以得到一个Principal对象来代表当前用户,并确认该用户是否拥有某叫角色的权限.

Servlet请求拥有请求级别的属性,与前面提到的应用级别属性类似.请求级别的属性经常用来传递状态信息到可视化组件(如JSP)。

Servlet容器保证被Servlet处理的请求处于单线程内.因此你不必担心在访问request对象的属性时会有多线程的问题。

（四）Servlet响应

Servlet的功能就是接收请求,然后生成相应的响应. 这是通过调用javax.servlet.http.HttpServletResponse的方法实现的.

1、设置头：你可以设置包含在响应中的头.最重要的头就是Content-Type,它用来告诉客户端内容的格式,比如:text/html代表html,text/xml代表XML。

2、设置Cookies　你可以加入cookie到当前的响应中。

3、发送错误响应　 你可以使用sendError()发送一个HTTP错误编号。

4、重定向到其他资源　你可以使用sendRedirect()定向到另外一个URL。

使用Response API 的一个最重要的原则就是:操作HEADer,Cookie的任何方法必须在第一次输出缓冲区满且发送到客户端前调用。

（五）、过滤

如果你的Servlet容器是基于Servlet规范2.3或更高,那么你就可以使用javax.servlet.Filter来对请求和响应作些处理.许多filter聚合在一起,每一个filter都有机会来对请求和响应做处理。

（六）、Servlet中的Session

HTTP的一个关键特性就是无状态,因此我们不知道某个请求是否是来自同一用户的请求,这将会使跨请求的会话变得很艰难。

为了解决这个问题,Servelet实现了一个javax.servlet.http.HttpSession.servlet容器将采用Cookie或URL Rewriting来保证接邻近的请求包含session id来标识该请求是同一个session。因此保存在session中的属性可以被多个请求共享。

为了不浪费资源,Session有一个可配置的超时时间设置.如果两个请求间的时间差超过了超时时间间隔,那么session中的数据会失效.你可以定义一个默认的超时时。

Servlet例子：

**参见：WebApp2项目中的包Servlet/myServlet.java**

<http://localhost:8084/webApp2/myServlet>

其他参考：ServletExamples项目的执行效果

* 1. 框架简介

常用框架：struct2、hiberate、spring

中文：Struct2框架简介：[RefBooks\Struts2入门v3.0.pdf](file:///E:\zsy\教学\Internet\RefBooks\Struts2入门v3.0.pdf)

学习视频<http://v.youku.com/v_playlist/f4511408o1p0.html>

原版：E:\zsy\教学\Internet\RefBooks\struts-2.2.3\docs\WW\tutorials.html

**重点讲解：Struct2框架简介， Hello World Using Struts 2**

E:\zsy\教学\Internet\RefBooks\struts-2.2.3\docs\WW\hello-world-using-struts-2.html

介绍了如何建立一个基本的HelloWorld应用程序，基于MVC架构的应用程序，Step by Step，然后介绍：How the Code Works，最后是：What to Remember。

E:\zsy\教学\Internet\RefBooks\struts-2.2.3\docs\WW\coding-struts-2-actions.html

介绍了如何在struts2中进行编程。

上机实验 8**：调试两个例程，理解MVC框架。**

**提问：MVC架构和JSP程序的区别，举例说明优点。**

* 1. 延伸学习

其他框架

JSP访问数据库

JSP案例：[RefBooks\JSP案例.doc](file:///E:\zsy\教学\Internet\RefBooks\JSP案例.doc)

1. B/S开发其他技术简介
   1. .Net技术
      1. .net简介

.NET是基于框架的一系列编程技术、服务器技术和开发环境的总称。

**.NET 框架包括 3 个主要的部分：**

编程语言：

* C# (Pronounced C sharp)
* Visual Basic (VB .NET)
* J# (Pronounced J sharp)

服务器技术和客户端技术：

* ASP .NET (Active Server Pages)
* Windows Forms (Windows desktop solutions)
* Compact Framework (PDA / Mobile solutions)

开发环境：

* Visual Studio .NET (VS .NET)
* Visual Web Developer

.NET框架类似于JRE，框架是对语言中立的，Visual Studio.net将C#，VB.net J#等翻译成中间语言进行执行，称为托管代码。C++是非托管代码。

框架通过类库提供各种最新技术的支持。

参见：.NET Framework 类库

.net <http://msdn.microsoft.com/zh-cn/library/ms229335(v=VS.90).aspx>

例如，要进行数据访问，需要了解和学习[System.Data](http://msdn.microsoft.com/zh-cn/library/system.data(v=VS.90).aspx) 类库，该类库包含组成大部分 ADO.NET 结构的类。ADO.NET 结构使您可以生成可用于有效管理来自多个数据源的数据的组件。

* + 1. .NET ASP

能够在服务器端集成编程语言的代码，极大的扩充了服务器端的功能，利用框架提供的编程能力。

例如：webApp.net项目，在.net asp项目中，每个网页都带有两个附件。一是.cs附件，一是designer.cs附件，后一个是设计器自己生成，前一个储存用户代码。这两个附件组成了服务器端代码部分。使用partial关键字，即分部类型，该类型定义允许将类、结构或接口的定义拆分到多个文件中。

Asp.net需要进行编译后执行，编译时编译器将代码全部转换为中间语言。

例如：下面是login.aspx文件的文件头。

<%@ Page Title="登Ì?录?" Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true"

CodeBehind="Login.aspx.cs" Inherits="WebApp.net.Account.Login" %>

演示实验 6**：在该文件中加入按钮，并且加入代码，然后进行执行。**

问题：每次服务器端代码的执行，网页都要刷新一遍，如何解决。

* 1. SOA 及 Web Service 技术

下次文档补充。

1. Android程序开发
2. 后记

道、术与计算机编程语言的学习

“以道御术”意为：以道义来承载智术。道是境界、修养，术是智力、技巧。悟道比炼术更高一筹。

在计算机编程的学习中，要充分把握好道和术的关系。他们是相对的。

相对于做人和专业学习，做人为道，专业为术。

相对于专业基础和专业技术，基础为道，技术为术。基础使人具备专业素养，能够理解和贯通专业的本质。

相对于编程技术和编程语言，技术为道，语言为术。

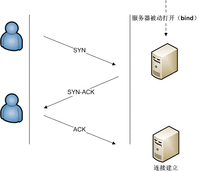
学习，首先悟道，其次炼术，道不通，则术不达。计算机的学习中，尤其重道，否则，出了问题不知如何去解决。

本门课程讲的是Internet编程技术，因涉及面比较广，所以，以道理讲解为主，关于技术的练习，实验课的时间是不够的，需要自己掌握。

附录1：TCP协议参考

应用层向TCP层发送用于网间传输的、用8位字节表示的数据流，然后TCP把数据流分割成适当长度的报文段（通常受该计算机连接的网络的数据链路层的最大传送单元（MTU）的限制）。之后TCP把结果包传给IP层，由它来通过网络将包传送给接收端实体的TCP层。TCP为了保证不发生丢包，就给每个字节一个序号，同时序号也保证了传送到接收端实体的包的按序接收。然后接收端实体对已成功收到的字节发回一个相应的确认（ACK）；如果发送端实体在合理的往返时延（RTT）内未收到确认，那么对应的数据（假设丢失了）将会被重传。TCP用一个校验和函数来检验数据是否有错误；在发送和接收时都要计算和校验。

[TCP](http://i5.qhimg.com/t01bebfaa9473e1a15e.gif)



首先，TCP建立连接之后，通信双方都同时可以进行数据的传输，其次，它是全双工的；在保证可靠性上，采用超时重传和捎带确认机制。

在流量控制上，采用滑动窗口协议，协议中规定，对于窗口内未经确认的分组需要重传。

在拥塞控制上，采用广受好评的TCP拥塞控制算法（也称AIMD算法），该算法主要包括三个主要部分：1，加性增、乘性减；2，慢启动；3，对超时事件做出反应。

1. 《*Software Design: From Programming to Architecture*》(Wiley 2003) [↑](#footnote-ref-1)
2. <http://www.ibm.com/developerworks/cn/webservices/ws-sca/> [↑](#footnote-ref-2)