基本常识

**1.软件生命周期(SDLC，Systems Development Life Cycle,SDLC)**是软件的产生直到报废或停止使用的生命周期。旧的解释是周期内有问题定义、可行性分析、总体描述、系统设计、编码、调试和测试、验收与运行、维护升级到废弃等阶段，这种按时间分程的思想方法是软件工程中的一种思想原则，即按部就班、逐步推进，每个阶段都要有定义、工作、审查、形成文档以供交流或备查，以提高软件的质量。

1.1 问题定义。要求系统分析员与用户进行交流，弄清“用户需要计算机解决什么问题”然后提出关于“系统目标与范围的说明”，提交用户审查和确认。

1.2 可行性研究。一方面在于把待开发的系统的目标以明确的语言描述出来，另一方面从经济、技术、法律等多方面进行可行性分析。

1.3 需求分析。弄清用户对软件系统的全部需求，编写需求规格说明书和初步的用户手册，提交评审。

1.4 开发阶段。开发阶段由三个阶段组成：，1.设计 2. 实现：根据选定的程序设计语言完成源程序的编码。 3. 测试

1.5 维护：维护包括四个方面

1.6 在软件设计完成后要经过严密的测试，以发现软件在整个设计过程中存在的问题并加以纠正。整个测试过程分单元测试、组装测试以及系统测试三个阶段进行。测试的方法主要有白盒测试和黑盒测试两种。在测试过程中需要建立详细的测试计划并严格按照测试计划进行测试，以减少测试的随意性。

1.7 软件维护是软件生命周期中持续时间最长的阶段。在软件开发完成并投入使用后，由于多方面的原因，软件不能继续适应用户的要求。要延续软件的使用寿命，就必须对软件进行维护。软件的维护包括纠错性维护和改进性维护两个方面。

**2 周期模型**

2.1瀑布模型: 瀑布模型通过强制性的要求提供规约文档来确保每个阶段都能很好的完成任务。但是实际上往往难以办到，因为整个的模型几乎都是以文档驱动的，这对于非专业的用户来说是难以阅读和理解的。

2.2迭代式模型: 在某种程度上，开发迭代是一次完整地经过所有工作流程的过程：（至少包括）需求工作流程、分析设计工作流程、实施工作流程和测试工作流程。实质上，它类似小型的瀑布式项目。RUP认为，所有的阶段（需求及其它）都可以细分为迭代。

2.3快速原型模型: 一般来说，根据客户的需要在很短的时间内解决用户最迫切需要，完成一个可以演示的产品。这个产品只是实现部分的功能（最重要的）。它最重要的目的是为了确定用户的真正需求。

**3 白盒测试：**白盒指的是盒子是可视的，你清楚盒子内部的东西以及里面是如何运作的。"白盒"法全面了解程序内部逻辑结构、对所有逻辑路径进行测试。"白盒"法是穷举路径测试。在使用这一方案时，测试者必须检查程序的内部结构，从检查程序的逻辑着手，得出测试数据。常用的软件测试方法有两大类：静态测试方法和动态测试方法

**4 黑盒测试**也称功能测试，是以用户的角度，从输入数据与输出数据的对应关系出发进行测试的。它是通过测试来检测每个功能是否都能正常使用。在测试中，把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，在程序接口进行测试，它只检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数据而产生正确的输出信息。黑盒测试着眼于程序外部结构，不考虑内部逻辑结构，主要针对软件界面和软件功能进行测试。

数据库

**1.开发流程：**

#region

//连接数据库的基本步骤

//1.连接字符串

String constr = "Data Source=DESKTOP-J4JFLTU\\SQLEXPRESS;Initial CataLog=MyFirstDataBase;Integrated Security=True";

//2.创建连接对象

using(SqlConnection con=new SqlConnection(constr))

{

//3.创建sql语句

string sql = "select \* from TblScore";

//4.创建命令对象

using(SqlCommand cmd=new SqlCommand(sql,con))

{ //5.打开数据库

con.Open();

//通过调用ExecuteReader()方法，将给定的sql语句在服务器端执行

//执行完毕后，服务器端就已经查询出了数据，但是数据是保存在数据库服务器的内存当中

//并没有返回给应用程序，只是返回给应用程序一个reader对象，这个对象就是用来获取数据的对象

//1.SqlDataReader只读，只进没法后退，使用时必须保证连接Opend()是打开连接状态

//2.Reader使用完毕时，必须关闭reader,释放，同时关闭连接对象

//3.Reader默认独占一个连接

//6.执行语句

using(SqlDataReader reader=cmd.ExecuteReader())

{

//接下来通过reader对象一条一条获取数据

//1.在获取数据之前，先判断下本次执行查询后，是否查询到了数据

if(reader.HasRows)//如果有数据，则为true

{

//2.如果有数据，那么接下来就一条一条获取数据

//3.reader指向第一条数据的前一条，需要调用Read()方法，向后移动一条数据，

//如果成功移动到某条数据，返回true,否则返回false

while(reader.Read())

{

//获取当前reader指向的每一行数据

//reader.FielCount，可以获取当前查询语句查询出的列的个数

//reader当遇到数据库中null值的时，通过reader.GetValue(i)或者索引获取的是事DBNull.Value.不是c#的null,

//而DBNull.Value的ToString()方法返回的是空字符串，所以不会报错

for(int i=0;i<reader.FieldCount;++i)

{

//可以通过索引或者列的名称获取列的值

Console.Write(reader[i]+"\t|\t"); //1.利用索引器读取数据

//Console.Write(reader["tName"] + "\t|\t"); //2.利用列的名称读取数据

//Console.Write(reader.GetValue(i) + "\t|\t"); //3.GetValue()只能通过索引来获取列的值

//reader.GetXxxxxx(); //如果要获取到对应数据库中的类型，要使用

}

Console.WriteLine();//换行

}

}

else

{

Console.WriteLine("没有查询到任何数据！！！");

}

}

｝7释放命令对象 此中using负责释放

Console.ReadKey();

｝8释放连接对象

#endregion

注意：

1. cmd.ExecuteNonQuery();//当执行insert\delete\update语句时,返回int类型，表示所影响的行数。特别注意ExecuteNonQuery()执行其他语句，永远返回-1

2．cmd.ExecuteReader();//当查询多行，多列结果的时候

3.cmd.ExecuteScalar();//当执行返回单个结果

**2. 数据抽象：**物理抽象、概念抽象、视图级抽象,内模式、模式、外模式

**3. SQL语言**包括数据定义、数据操纵(Data Manipulation),数据控制(Data Control)

数据定义：Create Table,Alter Table,Drop Table, Craete/Drop Index等

数据操纵：Select ,insert,update,delete,

数据控制：grant,revoke

**4. SQL常用命令：**

Select \* FROM Table\_name;//建视图

Create UNIQUE INDEX index\_name ON TableName(col\_name);//建索引

INSERT INTO tablename {column1,column2,…} values(exp1,exp2,…);//插入

INSERT INTO Viewname {column1,column2,…} values(exp1,exp2,…);//插入视图实际影响表

UPDATE tablename SET name=’zang 3’ condition;//更新数据condition为条件

DELETE FROM Tablename WHERE condition;//删除condition为条件

GRANT (Select,delete,…) ON (对象) TO USER\_NAME [WITH GRANT OPTION];//授权

REVOKE (权限表) ON(对象) FROM USER\_NAME [WITH REVOKE OPTION] //撤权

**5. 完整性约束**：实体完整性、参照完整性、用户定义完整性

**6. 事务**：是一系列的数据库操作，是数据库应用的基本逻辑单位。事务就是被绑定在一起作为一个逻辑工作单元的SQL语句分组，如果任何一个语句操作失败那么整个操作就被失败，以后操作就会回滚到操作前状态，或者是上有个节点。为了确保要么执行，要么不执行，就可以使用事，事务性质：原子性、一致性或可串性、隔离性、持久性

6.1原子性。即不可分割性，事务要么全部被执行，要么就全部不被执行

6.2一致性或可串性。事务的执行使得数据库从一种正确状态转换成另一种正确状态

6.3隔离性。在事务正确提交之前，不允许把该事务对数据的任何改变提供给任何其他事务，

6.4持久性。事务正确提交后，其结果将永久保存在数据库中，即使在事务提交后有了其他故障，事务的处理结果也会得到保存。

**7. 存储过程：**存储过程是一个预编译的SQL语句，优点是允许模块化的设计，就是说只需创建一次，以后在该程序中就可以调用多次。1 存储过程因为SQL语句已经预编绎过了，因此运行的速度比较快。2 可保证数据的安全性和完整性。通过存储过程可以使没有权限的用户在控制之下间接地存取数据库，从而保证数据的安全。通过存储过程可以使相关的动作在一起发生，从而可以维护数据库的完整性。3.可以降低网络的通信量。存储过程主要是在服务器上运行，减少对客户机的压力。4：存储过程可以接受参数、输出参数、返回单个或多个结果集以及返回值。可以向程序返回错误原因5：存储过程可以包含程序流、逻辑以及对数据库的查询。同时可以实体封装和隐藏了数据逻辑。

**8.触发器：** 当满足触发器条件，则系统自动执行触发器的触发体。触发时间：有before,after.触发事件：有insert,update,delete三种。触发类型：有行触发、语句触发，触发器是一中特殊的存储过程，主要是通过事件来触发而被执行的。它可以强化约束，来维护数据的完整性和一致性，可以跟踪数据库内的操作从而不允许未经许可的更新和变化。可以联级运算。如，某表上的触发器上包含对另一个表的数据操作，而该操作又会导致该表触发器被触发。

**9.内联接,外联接区别：**内连接是保证两个表中所有的行都要满足连接条件，在外连接中，某些不满条件的列也会显示出来，也就是说，只限制其中一个表的行，而不限制另一个表的行。分左连接、右连接、全连接三种

**10索引的作用？和它的优点缺点是什么？**索引就一种特殊的查询表，数据库的搜索引擎可以利用它加速对数据的检索。索引可以是唯一的，创建索引允许指定单个列或者是多个列。缺点是它减慢了数据录入的速度，同时也增加了数据库的尺寸大小。索引有B-TREE、BIT、CLUSTER等类型。ORACLE使用了一个复杂的自平衡B-tree结构;通常来说，在表上建立恰当的索引，查询时会改进查询性能。但在进行插入、删除、修改时，同时会进行索引的修改，在性能上有一定的影响。

**11什么叫视图？游标是什么？**视图是一种虚拟的表，具有和物理表相同的功能。可以对视图进行增，改，查，操作，试图通常是有一个表或者多个表的行或列的子集。对视图的修改不影响基本表。它使得我们获取数据更容易，相比多表查询。

游标：是对查询出来的结果集作为一个单元来有效的处理。游标可以定在该单元中的特定行，从结果集的当前行检索一行或多行。可以对结果集当前行做修改。一般不使用游标，但是需要逐条处理数据的时候，游标显得十分重要。

**12面试过程中多次碰到两道SQL查询的题目**，一是查询A(ID,Name)表中第31至40条记录，ID作为主键可能是不连续增长的列，完整的查询语句如下：select top 10 \* from A where ID >(select max(ID) from (select top 30 ID from A order by A ) T) order by A

另外一道题目的要求是查询表A中存在ID重复三次以上的记录,完整的查询语句如下：

select \* from(select count(ID) as count from table group by ID)T where T.count>3

**13简要叙述一下SQL Server 2000中使用的一些数据库对象吗?** 答案包括这样一些对象:表格、视图、用户定义的函数，以及存储过程;如果他们还能够提到像触发器这样的对象就更好了。

**14 NULL是什么意思?**NULL(空)这个值是数据库世界里一个非常难缠的东西， NULL这个值表示UNKNOWN(未知):它不表示“”(空字符串)。假设您的SQL Server数据库里有ANSI\_NULLS，当然在默认情况下会有，对NULL这个值的任何比较都会生产一个NULL值。您不能把任何值与一个 UNKNOWN值进行比较，并在逻辑上希望获得一个答案。您必须使用IS NULL操作符。

**15什么是主键?什么是外键?** 主键是表格里的(一个或多个)字段，只用来定义表格里的行;主键里的值总是唯一的。外键是一个用来建立两个表格之间关系的约束。这种关系一般都涉及一个表格里的主键字段与另外一个表格(尽管可能是同一个表格)里的一系列相连的字段。那么这些相连的字段就是外键。

**16你可以用什么来确保表格里的字段只接受特定范围里的值?** 您希望听到的回答是Check限制，它在数据库表格里被定义，用来限制输入该列的值。

**17列举几种表连接方式：**等连接（内连接）、非等连接、自连接、外连接（左、右、全）

一般的相等连接： select \* from a, b where a.id = b.id; 这个就属于内连接。

LEFT OUTER JOIN：左外关联 RIGHT OUTER JOIN：右外关联 FULL OUTER JOIN：全外关联

例子：

SELECT e.last\_name, e.department\_id, d.department\_name

FROM employees e

LEFT OUTER JOIN departments d

ON (e.department\_id = d.department\_id);

**18日志的作用是什么？**日志文件（Log File）记录所有对数据库数据的修改，主要是保护数据库以防止故障,以及恢复数据时使用。其特点如下：a)每一个数据库至少包含两个日志文件组。每个日志文件组至少包含两个日志文件成员。b)日志文件组以循环方式进行写操作。c)每一个日志文件成员对应一个物理文件。

**19 注入攻击：**用户输入的内容提交给服务器之后，服务器运行上面的代码构造出查询用户的SQL命令，但由于攻击者输入的内容非常特殊，所以最后得到的SQL命令变成：SELECT \* from Users WHERE login = '' or '1'='1' AND password = '' or '1'='1'。

如何防范SQL注入式攻击？1。用存储过程来执行所有的查询。SQL参数的传递方式将防止攻击者利用单引号和连字符实施攻击。此外，它还使得数据库权限可以限制到只允许特定的存储过程执行，所有的用户输入必须遵从被调用的存储过程的安全上下文，这样就很难再发生注入式攻击了。2.替换单引号，即把所有单独出现的单引号改成两个单引号，防止攻击者修改SQL命令的含义。再来看前面的例子，"SELECT \* from Users WHERE login = ''' or ''1''=''1' AND password = ''' or ''1''=''1'"显然会得到与"SELECT \* from Users WHERE login = '' or '1'='1' AND password = '' or '1'='1'"不同的结果。

计算机网络

**1. 协议的体系结构**

OSI分层（7层）：物理层、数据链路层、网络层、传输层、会话层、表示层、应用层。

TCP/IP分层（4层）：网络接口层、网际层、运输层、应用层。

五层协议（5层）：物理层、数据链路层、网络层、运输层、应用层。

*1.1 物理层*：激活、维持、关闭通信端点之间的机械特性、电气特性、功能特性以及过程特性。该层为上层协议提供了一个传输数据的物理媒体。

*1.2 数据链路层*：数据链路层在不可靠的物理介质上提供可靠的传输。该层的作用包括：物理地址寻址（MAC帧，半双工通信，CSMA/CD协议，即多点接入，载波监听，碰撞检测）、数据的成帧（ppp帧，全双工，点对点）、流量控制、数据的检错、重发等。

*1.3 网络层*：网络层负责对子网间的数据包进行路由选择。此外，网络层还可以实现拥塞控制、网际互连等功能。

*1.4 传输层*：第一个端到端，即主机到主机的层次。传输层负责将上层数据分段并提供端到端的、可靠TCP的或不可靠UDP的传输。此外，传输层还要处理端到端的差错控制和流量控制问题。

会话层：会话层管理主机之间的会话进程，即负责建立、管理、终止进程之间的会话。会话层还利用在数据中插入校验点来实现数据的同步。

表示层：表示层对上层数据或信息进行变换以保证一个主机应用层信息可以被另一个主机的应用程序理解。表示层的数据转换包括数据的加密、压缩、格式转换等。

*1.5 应用层*：为操作系统或网络应用程序提供访问网络服务的接口。

**2. IP地址的分类：**

2.1分类IP，

A类地址：以0开头，第一个字节范围：0~127；

B类地址：以10开头，第一个字节范围：128~191；

C类地址：以110开头，第一个字节范围：192~223；

D类地址：以1110开头，第一个字节范围为224~239；

2.2划分子网：二级ip到三级IP增加子网字段（从主机号中画出来的）；单位内部的事，对外表现出一个网络；有子网掩码；路由表包括目的网络地址，子网掩码，下一跳地址

2.3无分类编址CIDR:又变为二级ip；使用“斜线法/” ；可以划分为多个C类地址，又叫超网；最长前缀匹配，二叉线索查找路由。

3. ARP协议的工作原理

首先，每台主机都会在自己的ARP缓冲区中建立一个 ARP列表，以表示IP地址和MAC地址的对应关系。当源主机需要将一个数据包要发送到目的主机时，会首先检查自己 ARP列表中是否存在该 IP地址对应的MAC地址，如果有，就直接将数据包发送到这个MAC地址；如果没有，就向本地网段发起一个ARP请求的广播包，查询此目的主机对应的MAC地址。此ARP请求数据包里包括源主机的IP地址、硬件地址、以及目的主机的IP地址。网络中所有的主机收到这个ARP请求后，会检查数据包中的目的IP是否和自己的IP地址一致。如果不相同就忽略此数据包；如果相同，该主机首先将发送端的MAC地址和IP地址添加到自己的ARP列表中，如果ARP表中已经存在该IP的信息，则将其覆盖，然后给源主机发送一个 ARP响应数据包，告诉对方自己是它需要查找的MAC地址；源主机收到这个ARP响应数据包后，将得到的目的主机的IP地址和MAC地址添加到自己的ARP列表中，并利用此信息开始数据的传输。如果源主机一直没有收到ARP响应数据包，表示ARP查询失败。

**3.路由设备与相关层**

3.1 物理层：

中继器（Repeater，也叫放大器），集线器（使用CSMA/CD协议）。

3.2 数据链路层：

网桥：有多个硬件地址，使用CSMA/CD协议,转发表自学习，增大吞吐量，扩大物理范围，可互联不同物理层，不同速率。

交换机：多接口的网桥，全双工，可实现虚拟局域网(逻辑分组)VLAN。

3.3 网络层：

路由器：有多个硬件和IP地址，可以连接两个不同的网络，根据Ip地址进行路由选择

3.4 网关：网络层以上的设备。

**4 常见的路由选择协议，以及它们的区别**

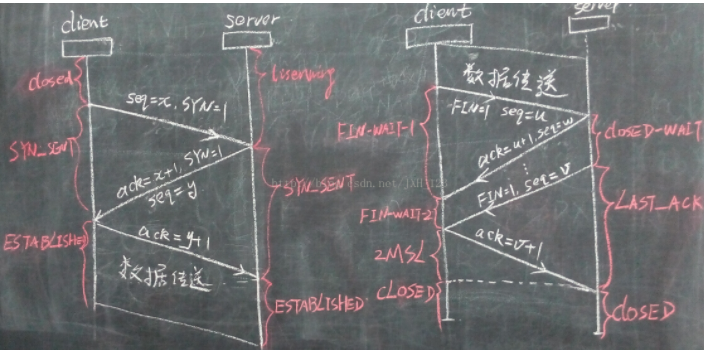
常见的路由选择协议有：RIP协议、OSPF协议。RIP（路由信息协议）：基于距离向量的路由选择协议，它选择路由的度量标准（metric)是跳数，最大跳数是15跳，如果大于15跳，它就会丢弃数据包。OSPF（开放最短路径优先）：底层是迪杰斯特拉算法，是链路状态路由选择协议，它选择路由的度量标准是带宽，延迟，可以知道“全网的拓扑结构图”，并以此利用迪杰斯特拉进行构建路由表。

**5.TCP与UDP的区别**

UDP是面向无连接的，不可靠的数据报服务，尽最大努力交付，面向报文；没有拥塞控制；支持一对多，多对多等方式；

TCP是面向连接，可靠的字节流服务，全双工，面向字节流；流量控制，拥塞控制; 点对点。可靠是通过滑动窗口顺序编号和确认实现的。

**6. TCP三次握手和四次挥手的全过程**



**7. TCP/IP中，每一层对应的协议**

链路层：ppp协议，CSMA/CD协议

网络层：IP协议、ICMP协议、ARP协议、RARP协议。

传输层：UDP协议、TCP协议。

应用层：FTP（文件传送协议）、Telenet（远程登录协议）、DNS（域名解析协议）、SMTP（邮件传送协议），POP3协议（邮局协议），HTTP协议。

TCP对应的协议：

（1 FTP：定义了文件传输协议，使用21端口。常说某某计算机开了FTP服务便是启动了文件传输服务。下载文件，上传主页，都要用到FTP服务。

（2 Telnet：它是一种用于远程登陆的端口，用户可以以自己的身份远程连接到计算机上，通过这种端口可以提供一种基于DOS模式下的通信服务。如以前的BBS是-纯字符界面的，支持BBS的服务器将23端口打开，对外提供服务。

（3 SMTP：定义了简单邮件传送协议，现在很多邮件服务器都用的是这个协议，用于发送邮件。如常见的免费邮件服务中用的就是这个邮件服务端口，所以在电子邮件设置-中常看到有这么SMTP端口设置这个栏，服务器开放的是25号端口。

（4 POP3：它是和SMTP对应，POP3用于接收邮件。通常情况下，POP3协议所用的是110端口。也是说，只要你有相应的使用POP3协议的程序（例如Fo-xmail或Outlook），就可以不以Web方式登陆进邮箱界面，直接用邮件程序就可以收到邮件（如是163邮箱就没有必要先进入网易网站，再进入自己的邮-箱来收信）。

（5 HTTP协议：是从Web服务器传输超文本到本地浏览器的传送协议。

UDP对应的协议：

（1 DNS：用于域名解析服务，将域名地址转换为IP地址。DNS用的是53号端口。

（2 SNMP：简单网络管理协议，使用161号端口，是用来管理网络设备的。由于网络设备很多，无连接的服务就体现出其优势。

（3 TFTP(Trival File Transfer Protocal)，简单文件传输协议，该协议在熟知端口69上使用UDP服务。

**8. 特殊的IP地址**

网络地址+主机地址全为零：0.0.0.0，已经不是一个真正意义上的IP地址了。它表示的是这样一个集合：所有不清楚的主机和目的网络。这里的“不清楚”是指在本机的路由表里没有特定条目指明如何到达。对本机来说，它就是一个“收容所”，所有不认识的“三无”人员，一律送进去。

A类网除去第一个字节网络地址全为1，回环测试 例如127.0.0.1代表“我自己”寻址这样一个地址，是不能把它发到网络接口的。除非出错，否则在传输介质上永远不应该出现目的地址为“127.0.0.1”的数据包。

主机号全0：指为单个网络地址；主机号全为1：这个网络上的所有主机，也叫广播地址。

组播地址：D类地址以1110开头，地址范围是224.0.0.0~239.255.255.255，D类地址作为组播地址（一对多的通信）；只有A,B,C有网络号和主机号之分，D类地址和E类地址没有划分网络号和主机号。

受限的广播地址：255.255.255.255 该IP地址指的是受限的广播地址。受限广播地址与一般广播地址（直接广播地址）的区别在于，受限广播地址之只能用于本地网络，路由器不会转发以受限广播地址为目的地址的分组；一般广播地址既可在本地广播，也可跨网段广播。例如：主机192.168.1.1/30上的直接广播数据包后，另外一个网段192.168.1.5/30也能收到该数据报；若发送受限广播数据报，则不能收到。

**9 差错检测？**

答：循环冗余检验 CRC ，计算出的结果叫做帧检验序列 FCS 。循环冗余检验序列 CRC 差错检测技术只能做到无差错接受，即凡是接收端数据链路层接受的帧，我们都能以非常接近于 1 的概率认为这些帧在传输过程中没有产生差错，但是要做到可靠传输（即发送什么就收到什么），也就是说，传输到接收端的帧无差错、无丢失、无重复，同时还按发送的顺序接收，这时就必须再加上确认和重传机制。

**10. 实现可靠传输的协议？**

1.停止等待协议：每发送完一帧就停止发送，直到收到接收到发送回来的确认在发送下一帧，如果没有收到接收端的确认，则通过设定的定时器超时了重传上一帧。其存在的三种可能：

重传可能会导致接收端收到相同的帧，这时候根据序号来判定，如果收到的帧的序号之前已经被接收到了，则新接收到的帧被丢弃。因为可能会出现接收端不能在一次情况就能正确接收，因此帧需要在发送端备份一份，直到被确认后才丢弃，因为该协议一次只能发送一帧，因此发送端的缓存区不需要太大。

2.连续 ARQ 协议：发送窗口大于1 ，接收窗口等于1 ，因此发送窗口已经发送到了序号为5 的帧，但是接收端接收到序号为3的帧出现错误时，那 3号以后的帧都需要重传，因此出现错误的情况可能会导致重传多个帧，同时为了能够在出错时重传，因此发送出来还没有经过确认的帧都需要在发送端全缓区进行保存，这种情况需要的缓冲区比停止等待协议需要的更大。但采用 n 比特来表示编号时，则发送窗口的的大小为时，该协议才能正确工作。若用n比特编号时，则发送窗口的大小 WT<=2n -1。

3.选择重传ARQ协议：发送窗口和接收窗口都大于 1 ，这种情况可能减少重传帧的数量，若用 n 比特编号时，则接收窗口的大小为 WR£ 2n /2 。

**11. 数据链路层协议可能提供的服务？**

答：成帧、链路访问、透明传输、可靠交付、流量控制、差错检测、差错纠正、半双工和全双工。最重要的是帧定界（成帧）、透明传输以及差错检测。其中，帧定界就是确定帧的界限，其方法有：字节计数法、字符填充法、零比特填充法；即应能传输任何的数据，在帧定界中用到的标记帧起点和结束的字符也应该能正确的被传输。

**12. 网络接口卡（网卡）的功能，适配器。**

答：（ 1 ）进行串行 / 并行转换。（ 2 ）对数据进行缓存。（ 3 ）在计算机的操作系统安装设备驱动程序。（ 4 ）实现以太网协议。

**13 路由表是做什么用的？在 Linux 环境中怎么配置一条默认路由？**

答： 路由表是用来决定如何将一个数据包从一个子网传送到另一个子网的，换句话说就是用来决定从一个网卡接收到的包应该送到哪一个网卡上去。路由表的每一行至少 有目标网络号、子网掩码、到这个子网应该使用的网卡这三条信息。当路由器从一个网卡接收到一个包时，它扫描路由表的每一行，用里面的子网掩码与数据包中的 目标 IP 地址做逻辑与运算（ & ）找出目标网络号。如果得出的结果网络号与这一行的网络号相同，就将这条路由表六下来作为备用路由。如果已经有备用路由了，就载这两条路由里将网络号最长的留下来，另一条丢掉（这是用无分类编址 CIDR 的情况才是匹配网络号最长的，其他的情况是找到第一条匹配的行时就可以进行转发了 ）。如此接着扫描下一行直到结束。如果扫描结束仍没有找到任何路由，就用默认路由。确定路由后，直接将数据包送到对应的网卡上去。在具体的实现中，路由表可能包含更多的信息为选路由算法的细节所用。在 linux 上可以用“ route add default gw< 默认路由器 IP> ”命令配置一条默认路由。

14 每个路由器在寻找路由时需要知道哪 5 部分信息？

答：目的地址：报文发送的目的地址；邻站的确定：指明谁直接连接到路由器的接口上； 路由的发现：发现邻站知道哪些网络；选择路由：通过从邻站学习到的信息，提供最优的到达目的地的路径；保持路由信息：路由器保存一张路由表，它存储所知道的所有路由信息。

**14 两台笔记本电脑连起来后 ping 不同，你觉得可能存在哪些问题？**

答：1）首先考虑是否是网络的问题2）局域网设置问题，电脑互联是要设置的。看是否安装了必要的网络协议，最重要的是 IP 地址是否设置正确。3）网卡驱动未安装正确4）防火墙设置有问题 5）是否有什么软件阻止了 ping 包

**15 常用计算机网络命令**

Ping ip ：测试两台主机是否连通

Traceroute ip/Tracert ip(window) ：追踪源点到目的点路由器的通断

**16. MTU（最大传输单元）？ MSS（最大分段大小）? 路径MTU?**

以太网(Ethernet)数据帧的数据部分的长度必须在46-1500字节之间,这是由以太网的物理特性决定的.这个1500字节被称为链路层的MTU(最大传输单元).但这并不是指链路层的长度被限制在1500字节,其实这个MTU指的是链路层的数据区.并不包括链路层的首部和尾部的18个字节.所以,事实上,这个1500字节就是网络层IP数据报的长度限制,这也就是说IP数据报大于1500字节,大于MTU.这个时候发送方IP层就需要分片(fragmentation).把数据报分成若干片,使每一片都小于MTU.而接收方IP层则需要进行数据报的重组. 这样就会多做许多事情,而更严重的是,由于UDP的特性,当某一片数据传送中丢失时,接收方便无法重组数据报.将导致丢弃整个UDP数据报。因此,在普通的局域网环境下，我建议将UDP的数据控制在1472字节以下为好.

以太网MAC帧

UDP 包的MTU大小就应该是1500(链路层帧) - IP头(20) - UDP头(8) = 1472(BYTES)

TCP 包的MTU大小就应该是 1500(链路层帧) - IP头(20) - TCP头(20) = 1460(BYTES)

MTU=MSS+TCP Header+IP Header.

MSS选项占4byte。MSS是每一个TCP报文段中数据字段的最大长度，注意：只是数据部分的字段，不包括TCP的头部。而Internet上标准的MTU（最小的MTU，链路层网络为x2.5时）为576，那么如果不设置，则MSS的默认值就为536个字节。很多时候，MSS的值最好取512的倍数。TCP报文段的分段与重组是在运输层完成的。*总结：UDP不会分段，就由IP来分。TCP会分段，当然就不用IP来分了！*

路径MTU: 当在同一个网络上的两台主机互相进行通信时，该网络的M T U是非常重要的。但是如果两台主机之间的通信要通过多个网络，那么每个网络的链路层就可能有不同的MTU。重要的不是两台主机所在网络的MTU的值，重要的是两台通信主机路径中的最小MTU。它被称作路径MTU。

操作系统

**1.什么是进程（Process）和线程（Thread）和程序？有何区别？**

•进程(process):

狭义定义：进程就是一段程序的执行过程。

广义定义：进程是一个具有独立功能的程序关于某个数据集合的一次运行活动。他是操作系统动态执行的基本单元，在传统的操作系统中,进程即是基本的分配单元，也是基本的执行单元。

1）进程是一个实体，每个进程都有自己的地址空间，一般情况下，包含文本区域、数据区域、堆栈，进程是系统进行资源分配和调度的一个独立单位，创建、撤销、切换系统开销大

2）进程是执行中的程序，程序是一个没有生命的实体，只有处理器赋予程序生命时，他才能成为一个活动的实体，我们称之为进程，是操作系统对正在运行程序的一种抽象，

3）进程本身不会运行，是线程的容器。线程不能单独执行，必须组成进程

4）一个程序至少有一个进程，一个进程至少有一个线程

5）对于操作系统来讲，一个任务就是一个进程，比如打开一个浏览器就是启动一个浏览器进程。打开一个记事本就是启动一个记事本进程，打开2个记事本就是启动2个记事本进程

6）有些进程还不止同时做一件事情。比如打开word，他可以同时进行打字、打印、拼音检查等等，在一个进程内部，要同时干多件事情。

进程状态：

1）就绪：获取出CPU外的所有资源、只要处理器分配资源就可以马上执行

2）运行：获得处理器分配的资源，程序开始执行

3）阻塞：当程序条件不够的时候，需要等待提交满足的时候才能执行。

•线程：如果说操作系统引入进程的目的是为了提高程序并发执行，以提高资源利用率和系统吞吐量。那么操作系统中引入线程的目的，则是为了减少进程并发执行过程中所付出的时空开销，使操作系统能很好的并发执行。

1）一个进程中至少有一个线程，不然就没有存在的意义，每个线程都运行在进程的上下文中，并共享相同的代码和全局数据。仅保存少量寄存器内容，系统开销小。

2）在一个进程内部，要同时干多件事情，就需要同时运行多个子任务，我们把进程内的这些子任务叫做线程

3）多线程就是为了同步完成多项任务(在单个程序中同时运行多个线程完成不同的任务和工作)，不是为了提高运行效率，而是为了提高资源使用效率来提高系统的效率

4）一个简单的比喻，多线程就像是火车上的每节车厢，而进程就是火车

5）线程是程序执行流的最小单元。线程是进程的一个实体，是CPU调度和分派的基本单位，一个标准的线程由当前的线程ID、当前指令指针、寄存器和堆栈组成

6）同一个进程中的多个线程之间可以并发执行

线程状态：

1）就绪：指线程具备运行的所有条件，逻辑上可以运行，在等待处理机

2）运行：指线程占用处理机正在运行

3）阻塞：线程在等待一个事件，逻辑上不可执行

*如果我们要同时执行多个任务怎么办？*

1）启动多个进程，每个进程虽然只有一个线程，但是多个进程可以一块执行多个任务

2）启动一个进程，在一个进程内启动多个线程，这样多个线程也可以一块执行多个任务

　　进程与应用程序的区别在于：应用程序作为一个静态文件存储在计算机系统的硬盘等存储空间中，程序不能申请系统资源，不能被系统调度，也不能作为独立运行的单位，因此，它不占用系统的运行资源。而进程则是处于动态条件下由操作系统维护的系统资源管理实体。是执行中程序的一个实例。进程是资源申请、调度和独立运行的单位，因此，它使用系统中的运行资源

**2.进程的五状态模型：理论上有：**

运行态：该进程正在执行。

就绪态：进程已经做好了准备，只要有机会就开始执行。

阻塞态（等待态）：进程在某些事情发生前不能执行，等待阻塞进程的事件完成。

新建态：刚刚创建的进程，操作系统还没有把它加入到可执行进程组中，通常是进程控制块已经创建但是还没有加载到内存中的进程。

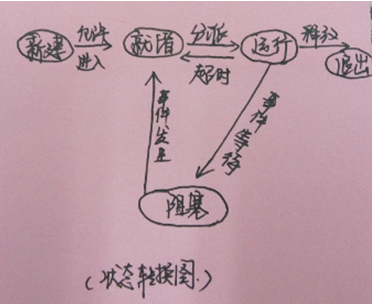
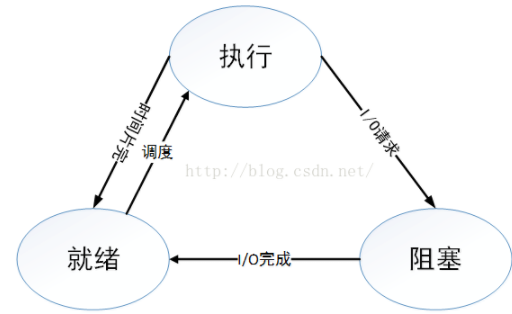
退出态：操作系统从可执行进程组中释放出的进程，或由于自身或某种原因停止运行。

操作系统上说：从程序员的角度，我们认为进程有三种状态，包括运行，停止，终止。

以下两种状态是不可能发生的：

阻塞——>运行：即使给阻塞进程分配CPU，也无法执行，操作系统在进行调度时不会从阻塞队列进行挑选，而是从就绪队列中选取

就绪——>阻塞：就绪态根本就没有执行，谈不上进入阻塞态。

**3. Windows下的内存是如何管理的？**

Windows提供了3种方法来进行内存管理：1.虚拟内存，最适合用来管理大型对象或者结构数组；2.内存映射文件，最适合用来管理大型数据流（通常来自文件）以及在单个计算机上运行多个进程之间共享数据；3。内存堆栈，最适合用来管理大量的小对象。

*Windows操纵内存可以分两个层面：物理内存和虚拟内存。*

其中物理内存由系统管理，不允许应用程序直接访问，应用程序可见的只有一个2G地址空间，而内存分配是通过堆进行的。对于每个进程都有自己的默认堆，当一个堆创建后，就通过虚拟内存操作保留了相应大小的地址块（不占有实际的内存，系统消耗很小）。当在堆上分配一块内存时，系统在堆的地址表里找到一个空闲块（如果找不到，且堆创建属性是可扩充的，则扩充堆大小），为这个空闲块所包含的所有内存页提交物理对象（在物理内存上或硬盘的交换文件上），这时就可以访问这部分地址。提交时，系统将对所有进程的内存统一调配，如果物理内存不够，系统试图把一部分进程暂时不访问的页放入交换文件，以腾出部分物理内存。释放内存时，只在堆中将所在的页解除提交（相应的物理对象被解除），继续保留地址空间。如果要知道某个地址是否被占用/可不可以访问，只要查询此地址的虚拟内存状态即可。如果是提交，则可以访问。如果仅仅保留，或没保留，则产生一个软件异常。此外，有些内存页可以设置各种属性。如果是只读，向内存写也会产生软件异常。

**4. Windows消息调度机制是**：消息队列 处理消息队列的顺序。首先Windows绝对不是按队列先进先出的次序来处理的，而是有一定优先级的。优先级通过消息队列的状态标志来实现的。首先，最高优先级的是别的线程发过来的消息（通过sendmessage）；其次，处理登记消息队列消息；再次处理QS\_QUIT标志，处理虚拟输入队列，处理wm\_paint；最后是wm\_timer。

**5. 描述实时系统的基本特性：**所谓“实时操作系统”，实际上是指操作系统工作时，其各种资源可以根据需要随时进行动态分配。由于各种资源可以进行动态分配，因此，其处理事务的能力较强、速度较快。

**6. 中断和轮询的特点**

6.1对I/O设备的程序轮询的方式，是早期的计算机系统对I/O设备的一种管理方式。它定时对各种设备轮流询问一遍有无处理要求。程序轮询毕竟占据了CPU相当一部分处理时间，因此，程序轮询是一种效率较低的方式，在现代计算机系统中已很少应用。6.2程序中断通常简称中断，是指CPU在正常运行程序的过程中，由于预先安排或发生了各种随机的内部或外部事件，使CPU中断正在运行的程序，而转到为响应的服务程序去处理。轮询——效率低，等待时间很长，CPU利用率不高。中断——容易遗漏一些问题，CPU利用率高。

**7. 什么是临界区？如何解决冲突？**

每个进程中访问临界资源的那段程序称为临界区，每次只准许一个进程进入临界区，进入后不允许其他进程进入。（1）如果有若干进程要求进入空闲的临界区，一次仅允许一个进程进入；（2）任何时候，处于临界区内的进程不可多于一个。如已有进程进入自己的临界区，则其它所有试图进入临界区的进程必须等待；（3）进入临界区的进程要在有限时间内退出，以便其它进程能及时进入自己的临界区；（4）如果进程不能进入自己的临界区，则应让出CPU，避免进程出现“忙等”现象。

**8. 说说分段和分页**

8.1页是信息的物理单位，分页是为实现离散分配方式，提高内存的利用率；分页仅仅是由于系统管理的需要，而不是用户的需要。页的大小固定且由系统确定，把逻辑地址划分为页号和页内地址两部分，是由机器硬件实现的，因而一个系统只能有一种大小的页面。8.2段是信息的逻辑单位，它含有一组其意义相对完整的信息。分段的目的是为了能更好的满足用户的需要。段的长度却不固定，决定于用户所编写的程序，通常由编辑程序在对源程序进行编辑时，根据信息的性质来划分。

**9. 说出你所知道的保持进程同步的方法？**

进程间同步的主要方法有原子操作、信号量机制、自旋锁、管程、会合、分布式系统等。

**10. Linux中常用到的命令**

显示文件目录命令ls 如ls

改变当前目录命令cd 如cd /home

建立子目录mkdir 如mkdir xiong

删除子目录命令rmdir 如rmdir /mnt/cdrom

删除文件命令rm 如rm /ucdos.bat

文件复制命令cp 如cp /ucdos /fox

获取帮助信息命令man 如man ls

显示文件的内容less 如less mwm.lx

重定向与管道type 如type readme>>direct，将文件readme的内容追加到文direct中

**11. makefile文件的作用是什么？**

一个工程中的源文件不计其数，其按类型、功能、模块分别放在若干个目录中。makefile定义了一系列的规则来指定哪些文件需要先编译，哪些文件需要后编译，哪些文件需要重新编译，甚至于进行更复杂的功能操作。因为makefile就像一个Shell脚本一样，其中也可以执行操作系统的命令。makefile带来的好处就是——“自动化编译”。一旦写好，只需要一个make命令，整个工程完全自动编译，极大地提高了软件开发的效率。make是一个命令工具，是一个解释makefile中指令的命令工具。一般来说，大多数的IDE都有这个命令，比如：Delphi的make，Visual C++的nmake，Linux下GNU的make。可见，makefile都成为了一种在工程方面的编译方法。

**12. 你知道操作系统的内容分为几块吗？什么叫做虚拟内存？他和主存的关系如何？内存管理属于操作系统的内容吗？**

操作系统的主要组成部分：进程和线程的管理，存储管理，设备管理，文件管理。虚拟内存是一些系统页文件，存放在磁盘上，每个系统页文件大小为4K，物理内存也被分页，每个页大小也为4K，这样虚拟页文件和物理内存页就可以对应，实际上虚拟内存就是用于物理内存的临时存放的磁盘空间。页文件就是内存页，物理内存中每页叫物理页，磁盘上的页文件叫虚拟页，物理页+虚拟页就是系统所有使用的页文件的总和。

**13. 线程是否具有相同的堆栈？dll是否有独立的堆栈？**

13.1 每个线程有自己的堆栈，13.2 因为dll中的代码是被某些线程所执行，只有线程拥有堆栈。如果dll中的代码是exe中的线程所调用，那么这个时候这个dll没有独立的堆栈。如果dll中的代码是由dll自己创建的线程所执行，那么dll有独立的堆栈。

以上讲的是堆栈，如果对于堆来说，每个dll有自己的堆，所以如果是从dll中动态分配的内存，最好是从dll中删除；如果你从dll中分配内存，然后在exe中，或者另外一个dll中删除，很有可能导致程序崩溃。

**14. 什么是缓冲区溢出？有什么危害？其原因是什么？**

　14.1缓冲区溢出是指当计算机向缓冲区内填充数据时超过了缓冲区本身的容量，溢出的数据覆盖在合法数据上。14.2危害：在当前网络与分布式系统安全中，被广泛利用的50%以上都是缓冲区溢出，其中最著名的例子是1988年利用fingerd漏洞的蠕虫。而缓冲区溢出中，最为危险的是堆栈溢出，因为入侵者可以利用堆栈溢出，在函数返回时改变返回程序的地址，让其跳转到任意地址，带来的危害一种是程序崩溃导致拒绝服务，另外一种就是跳转并且执行一段恶意代码，比如得到shell，然后为所欲为。通过往程序的缓冲区写超出其长度的内容，造成缓冲区的溢出，从而破坏程序的堆栈，使程序转而执行其它指令，以达到攻击的目的。14.3造成缓冲区溢出的主原因是程序中没有仔细检查用户输入的参数。

**15. 什么是死锁？其条件是什么？怎样避免死锁？**

15.1死锁的概念：在两个或多个并发进程中，如果每个进程持有某种资源而又都等待别的进程释放它或它们现在保持着的资源，在未改变这种状态之前都不能向前推进，称这一组进程产生了死锁。通俗地讲，就是两个或多个进程被无限期地阻塞、相互等待的一种状态。死锁产生的原因主要是： 系统资源不足； 进程推进顺序非法。

15.2产生死锁的必要条件：（1）互斥（mutualexclusion），一个资源每次只能被一个进程使用；（2）不可抢占（nopreemption），进程已获得的资源，在未使用完之前，不能强行剥夺；（3）占有并等待（hold andwait），一个进程因请求资源而阻塞时，对已获得的资源保持不放；（4）环形等待（circularwait），若干进程之间形成一种首尾相接的循环等待资源关系。这四个条件是死锁的必要条件，只要系统发生死锁，这些条件必然成立，而只要上述条件之一不满足，就不会发生死锁。

15.3死锁的解除与预防：理解了死锁的原因，尤其是产生死锁的四个必要条件，就可以最大可能地避免、预防和解除死锁。所以，在系统设计、进程调度等方面注意如何不让这四个必要条件成立，如何确定资源的合理分配算法，避免进程永久占据系统资源。此外，也要防止进程在处于等待状态的情况下占用资源。因此，对资源的分配要给予合理的规划。死锁的处理策略：鸵鸟策略、预防策略、避免策略、检测与恢复策略。

**16 进程同步** 进程同步的主要任务：是对多个相关进程在执行次序上进行协调，以使并发执行的诸进程之间能有效地共享资源和相互合作，从而使程序的执行具有可再现性。同步机制遵循的原则：（1）空闲让进；（2）忙则等待（保证对临界区的互斥访问）；（3）有限等待（有限代表有限的时间，避免死等）；（4）让权等待，（当进程不能进入自己的临界区时，应该释放处理机，以免陷入忙等状态）。

**17. 进程间的通信是如何实现的？**

现在linux使用的进程间通信方式：（1）管道（pipe）和有名管道（FIFO）（2）信号（signal）（3）消息队列（4）共享内存（5）信号量（6）套接字（socket)

进程通信，是指进程之间的信息交换（信息量少则一个状态或数值，多者则是成千上万个字节）。因此，对于用信号量进行的进程间的互斥和同步，由于其所交换的信息量少而被归结为低级通信。

所谓高级进程通信指：用户可以利用操作系统所提供的一组通信命令传送大量数据的一种通信方式。操作系统隐藏了进程通信的实现细节。或者说，通信过程对用户是透明的。

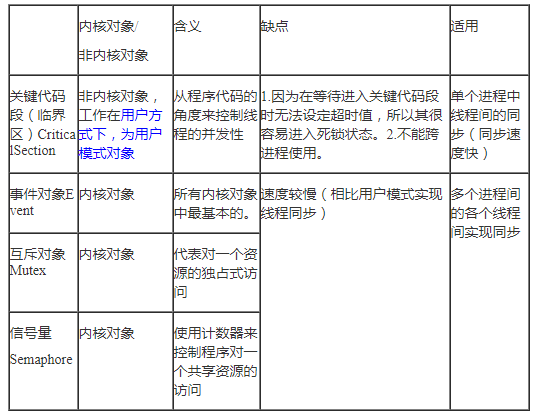
高级通信机制可归结为三大类：

（1）共享存储器系统（存储器中划分的共享存储区）；实际操作中对应的是“剪贴板”（剪贴板实际上是系统维护管理的一块内存区域）的通信方式，比如举例如下：word进程按下ctrl+c，在ppt进程按下ctrl+v，即完成了word进程和ppt进程之间的通信，复制时将数据放入到剪贴板，粘贴时从剪贴板中取出数据，然后显示在ppt窗口上。

（2）消息传递系统（进程间的数据交换以消息（message）为单位，当今最流行的微内核操作系统中，微内核与服务器之间的通信，无一例外地都采用了消息传递机制。应用举例：邮槽（MailSlot）是基于广播通信体系设计出来的，它采用无连接的不可靠的数据传输。邮槽是一种单向通信机制，创建邮槽的服务器进程读取数据，打开邮槽的客户机进程写入数据。

（3）管道通信系统（管道即：连接读写进程以实现他们之间通信的共享文件（pipe文件，类似先进先出的队列，由一个进程写，另一进程读））。实际操作中，管道分为：匿名管道、命名管道。匿名管道是一个未命名的、单向管道，通过父进程和一个子进程之间传输数据。匿名管道只能实现本地机器上两个进程之间的通信，而不能实现跨网络的通信。命名管道不仅可以在本机上实现两个进程间的通信，还可以跨网络实现两个进程间的通信。

**18．线程同步：**线程同步有四种方式：临界区、内核对象、互斥量、信号量。根据用户模式及内核模式下的同步方式的不同，分类及对比如下：由于进程同步产生了一系列经典的同步问题“生产者-消费者”问题，“哲学家进餐”问题，“读者-写者”问题。

**19 常见的操作系统使用的文件系统整理**

【FAT】增强的文件系统FAT32。同FAT16相比，FAT32主要具有以下特点：FAT32最大的优点是可以支持的磁盘大小达到32G，但是不能支持小于512MB的分区。由于采用了更小的簇，FAT32文件系统可以更有效率地保存信息。FAT32文件系统可以重新定位根目录和使用FAT的备份副本。

【NTFS】NTFS文件系统是一个基于安全性的文件系统，是Windows NT所采用的独特的文件系统结构，它是建立在保护文件和目录数据基础上，同时照顾节省存储资源、减少磁盘占用量的一种先进的文件系统。在NTFS分区上，可以为共享资源、文件夹以及文件设置访问许可权限。

【Ext2】是 GNU/Linux 系统中标准的文件系统，其特点为存取文件的性能极好，对于中小型的文件更显示出优势，这主要得利于其簇快取层的优良设计。

【Ext3】Ext3是一种日志式文件系统，是对ext2系统的扩展，它兼容ext2。日志式文件系统的优越性在于：它会将整个磁盘的写入动作完整记录在磁盘的某个区域上，以便有需要时可以回溯追踪。

**20 SOCKET编程**

*20.1 Socket接口*是TCP/IP网络的API，Socket接口定义了许多函数或例程，程序员可以用它们来开发TCP/IP网络上的应用程序。

网络的Socket数据传输是一种特殊的I/O，Socket也是一种文件描述符。Socket也具有一个类似于打开文件的函数调用Socket()，该函数返回一个整型的Socket描述符，随后的连接建立、数据传输等操作都是通过该Socket实现的。常用的Socket类型有两种：流式Socket（SOCK\_STREAM）和数据报式Socket（SOCK\_DGRAM）。流式是一种面向连接的Socket，针对于面向连接的TCP服务应用；数据报式Socket是一种无连接的Socket，对应于无连接的UDP服务应用。套接字接口(socket interface)是一组函数，它们和 Unix I/O 函数结合起来，用以创建网络应用。

*20.2 socket调用库函数主要有：*

客户端函数

int socket(int domain, int type, int protocol) 创建一个套接字描述符，仅仅部分打开，不能用于读写，type决定TCP或UDP

int connect(int sockfd, struct sockaddr \*serv\_addr, int addrlen); 建立和服务器的连接

服务端函数

Int bind(int sockfd,struct sockaddr \*my\_add, int addrlen) 将服务器套接字地址my\_add和套接字描述符sockfd联系起来

Int listen( int sockfd ,int bakelog) 将sockfd生成的主动套接字描述符转换为“监听套接字”描述符

建立服务器/客户端的连接 (面向连接TCP）

Int accept(int listenfd, struct sockaddr \*addr,int \*addrlen) 等待来自客户端连接请求到达“监听描述符”描述符，然后在add中填入客户端套接字地址等，并返回已连接描述符connfd

发送/接收数据

面向连接： send(sockid, buff, bufflen)

recv( )

面向无连接sendto(sockid,buff,…,addrlen)

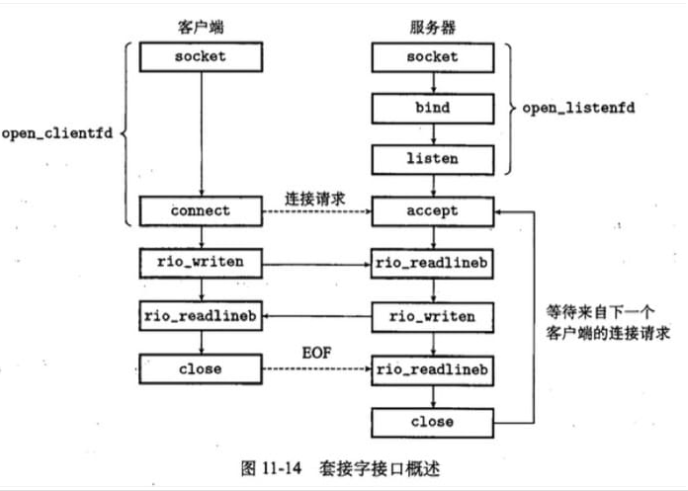
　　 recvfrom( )

释放套接字

close(sockid)

*20.3 客户端的工作流程：*首先调用socket函数创建一个clientfd描述符，仅仅部分打开，不能用于读写，然后调用connect函数将sockfd转换为“打开的套接字描述符clientfd”,可以进行读写了，并尝试与套接字地址为serv\_addr 的服务器建立因特网连接（其中serv\_addr是从主机中已存在的“HOSTS.TXT”文件根据域名读取的Ip地址和用户给的端口组成的结构体），connect函数会阻塞，直到连接成功或发生错误，直到此，客户端准备好，可以对“clientfd”进行读写通信了（一般用while进行读写）。使用完成用Close（clienfd）关闭套接字描述符。

*20.4 服务器的工作流程：*首先调用socket函数创建一个listenfd描述符，setsockopt函数进行配置服务器（此处没有描述）然后调用bind函数将my\_add（其中my\_add为结构体，使用包括INADDR\_ANY通配符地址（告诉内核接收到服务器的任何IP地址）和知名端口）与listenfd描述符联系起来，然后调用listen函数将“主动套接字描述符listenfd”转换为“监听套接字描述符listenfd”，再调用accept函数等待客户端的连接请求到达“监听套接字描述符listenfd”，并在addr中填入客户端的套接字地址等相关信息。返回一个“打开的已连接描述符connfd”，在客户端“打开的套接字描述符clientfd”和服务端的“打开的已连接描述符connfd”之间建立连接，至此，服务端可对“打开的已连接描述符connfd”进行读写通信了。



**21.web服务器**

Web 客户端和服务器之间的交互用的是一个基于文本的应用级协议，叫做 HTTP(Hypertext Transfer Protocol)。HTTP 是一个简单的协议。Web内容用HTML（超文本标记语言）的语言编写，特点1，告诉浏览器怎么显示各种文本和图像，2,包含指针指向存放在任何因特网上的内容。

Web 服务器以两种不同的方式向客户端提供内容，1.取一个磁盘文件，并将它的内容返回给客户端。磁盘文件称为静态内容(static content)，而返回文件给客户端的过程称为服务静态内容(serving static content)。2.运行一个可执行文件，并将它的输出返回给客户端。运行时可执行文件产生的输出称为动态内容(dynamic content)，而运行程序并返回它的输出到客户端的过程称为服务动态内容(serving dynamic content)。

URL(Universal Resource Locator)统一资源标识符

确定一个 URL 指向的是静态内容还是动态内容没有标准的规则。每个服务器对它所管理的文件都有自己的规则。一种常见方法是，确定一组目录，例如 cgi-bin，所有的可执行文件都必须存放这些目录中。”?”分割文件名和参数，”&”分割参数

后缀中的最开始的那个 / 不表示 Unix 的根目录。相反，它表示的是被请求内容类型的主目录。例如，可以将一个服务器配置成这样：所有的静态内容存放在目录 /usr/httpd/html 下。

最小的 URL 后缀是 / 字符，所有服务器将其扩展为某个默认的主页，例如 /index.html。这解释了为什么在浏览器中键入一个域名就可以取出一个网站的主页。浏览器在 URL 后添加缺失的 /，之后服务器把 / 扩展到某个默认的文件名。

HTTP/1.1定义了一些附加的报头为诸如缓存和安全等高级特性提供支持，还支持一种机制，允许客户端和服务器在同一条持久连接上执行多个事务。

其他论文项目

**1.图像评价指标：**

*1.1主观评价：*主要是通过观察者视觉感的好坏来评价图像质量的好坏，评价者可以从图像的整体画面是否清晰，是否保留了不同传感器的原始有用信息、融合图像中的细节边缘是否突出等几个角度来衡量图像融合的性能。

*1.2客观评价：*客观评价标准主要是通过建立一定的数学模型，通过计算机来定量地分析图像的各个特征信息。包含：平均梯度(AG)、相关系数(CC)、互信息(MI)、结构相似度(SSIM)、熵(IE)、空间频率(SF)。

1.2.1平均梯度(Average Gradient，AG)：AG能够很好地衡量融合后图像I\_F的模糊程度。AG值越大，则I\_F越清晰度与空间分辨率也越高。

1.2.2互信息(Mutual Information，MI):互信息可以度量融合图像I\_F与源图像I\_A、I\_B之间在灰度分布上的相关与相似性。其值越大，则相关性、相似度越高，I\_F保留I\_A、I\_B的信息量越多。计算如下：







其中，是图像的灰度级； 、是图像的联合概率密度；、、均表示图像的概率密度。

1.2.3信息熵(Information Entropy，IE)：IE能够很好地反映融合后图像I\_F中所包含信息量的丰富程度。其值越大，表示图像I\_F包含越丰富的细节等信息，融合质量也越高。



其中，为灰度级总数，一般为256；为中值为的像素总数与总像素数的比值，为图像灰度分布。

1.2.4 空间频率(Spatial Frequency，SF)

SF表示融合后图像I\_F的空间域整体活跃程度，图像质量的高低与SF值成正比。定义如下：







其中，I\_F (i,j)为测试的融合图像，RF、CF分别表示图像I\_F对应水平、垂直方向的空间频率。

**2．小论文融合**



**3. 链接库分为静态库，动态库，静态库；**

*3.1 DLL不是可执行文件*动态链接提供了一种方法，使进程可以调用不属于其可执行代码的函数。函数的可执行代码位于一个 DLL 中，该 DLL 包含一个或多个已被编译、链接并与使用它们的进程分开存储的函数。好处：有助于模块化，资源共享，节约空间内存。发展史上经历了“无库－静态链接库－动态链接库”的时代。

*3.2 区别：*静态库(static library)将函数和数据编译进一个二进制文件，通常可以命名为\*.lib（使用时，需要.h文件和.lib文件），编译器在链接过程中，将这些二进制数据复制出来，并与调用库的其他模块数据组合在一起，形成最终的可执行文件，等以后使用这个可执行文件时，就不需要这个静态库的支持了。动态链接库只有在需要的时候才会调用，所以相比静态链接库来说，有效减少程序运行时占用的内存空间。往往提供引人库（.lib）和一个(.dll),其中引人库(.lib)包含该DLL导出函数和变量的符号名，而(.dll)包含该DLL实际的函数和数据静态链接库与动态链接库都是共享代码的方式，如果采用静态链接库，则无论你愿不愿意，lib 中的指令都全部被直接包含在最终生成的 EXE 文件中了。但是若使用 DLL，该 DLL 不必被包含在最终 EXE 文件中，EXE 文件执行时可以“动态”地引用和卸载这个与 EXE 独立的 DLL 文件。静态链接库和动态链接库的另外一个区别在于静态链接库中不能再包含其他的动态链接库或者静态库，而在动态链接库中还可以再包含其他的动态或静态链接库。

*3.3常识概念：*对动态链接库：1. :只要遵循约定的DLL接口规范和调用方式，用各种语言编写的DLL都可以相互调用。2. 动态链接库随处可见，kernel32.dll中的函数主要处理内存管理和进程调度；user32.dll中的函数主要控制用户界面； gdi32.dll中的函数则负责图形方面的操作。3. VC动态链接库的分类：它们分别是Non-MFC DLL（非MFC动态库）导出函数为标准的C接口、MFC Regular DLL（MFC规则DLL）包含一个继承自CWinApp的类，但其无消息循环、MFC Extension DLL（MFC扩展DLL）采用MFC的动态链接版本创建，它只能被用MFC类库所编写的应用程序所调用。

*3.4库的调试与查看：*1.由于库文件不能单独执行，因而在按下F5（开始debug模式执行）或CTRL+F5（运行）执行时，其弹出对话框，要求用户输入可执行文 件的路径来启动库函数的执行。2.那就是将库工程和应用工程（调用库的工程）放置在同一VC工作区，只对应用工程进行调试，在应用工程调用库中函数的语 句处设置断点，执行后按下F11，这样就单步进入了库中的函数。3. 动态链接库中的导出接口可以使用Visual C++的Depends工具进行查看。

*3.5 声明导出函数：*1.方式1 extern "C" int \_\_declspec(dllexport) add(int x, int y);

2.方式2：LIBRARY语句说明.def文件相应的DLL；EXPORTS语句后列出要导出函数的名称。可以在.def文件中的导出函数名后加@n，表示要导出函数的序号为n

LIBRARY dllTest

EXPORTS

add @ 1

*3.6 DLL的调用方式* 1. 动态调用: 特点是完全由编程者用 API 函数加载和卸载 DLL，程序员可以决定 DLL 文件何时加载或不加载。由“LoadLibrary-GetProcAddress-FreeLibrary”系统Api提供的三位一体“DLL加载-DLL函数地址获取-DLL释放”方式。2. 静态调用：由编译系统完成对DLL的加载和应用程序结束时 DLL 的卸载(像智能指针shared\_ptr引用计数)。静态调用方式不再需要使用系统API来加载、卸载DLL以及获取DLL中导出函数的地址。因为当程序员通过静态链接方式编译生成应用程序时，应用 程序中调用的与.lib文件中导出符号相匹配的函数符号将进入到生成的EXE 文件中，.lib文件中所包含的与之对应的DLL文件的文件名也被编译器存储在 EXE文件内部。当应用程序运行过程中需要加载DLL文件时，Windows将根据这些信息发现并加载DLL，然后通过符号名实现对DLL 函数的动态链接。

#pragma comment(lib,"dllTest.lib") //告诉编译器与DLL相对应的.lib文件所在的路径及文件名(.lib中仅有DLL 导出函数的符号名及序号，参与编译)

extern "C" \_\_declspec(dllimport) add(int x,int y);// 声明导入函数

*3.7. 入口函数DllMain函数*：Windows在找不到DllMain的时候，系统会从其它运行库中引入一个不做任何操作的缺省DllMain函数版本。Windows必须查找并执行DLL里的DllMain函数作为加载DLL的依据，它使得DLL得以保留在内存里。这个函数并不属于导出函数，而是DLL的内部函数。

*3.8. \_\_stdcall约定*：如果通过VC++编写的DLL欲被其他语言编写的程序调用，应将函数的调用方式声明为\_\_stdcall方式，WINAPI都采用这种方式，而C/C++缺省的调用方式却为\_\_cdecl。\_\_stdcall方式与 \_\_cdecl对函数名最终生成符号的方式不同。

int \_\_stdcall add(int x, int y);// 在lib.h中，应这样声明add函数：

typedef int(\_\_stdcall \*lpAddFun)(int, int);// 在应用工程中函数指针类型应定义

*3.9 导出类(技巧)：*在应用工程中没有定义DLL\_FILE，故其包含point.h ；class \_declspec(dllimport) point //导入类point

//文件名：point.h，point类的声明

#ifndef POINT\_H

#define POINT\_H

#ifdef DLL\_FILE

Class \_declspec(dllexport) point //导出类point，这样就可同时被dll编译(dllexport导出类)和应用程序(dllimport导入类)编译使用

#else

class \_declspec(dllimport) point //导入类point，这样就可同时被dll编译和应用程序编译使用

#endif

{

public:

float y;

float x;

point(float x\_coordinate, float y\_coordinate);

};

#endif

//文件名：point.cpp，point类的实现

#ifndef DLL\_FILE

#define DLL\_FILE

#endif

#include "point.h"

//类point的构造函数

point::point(float x\_coordinate, float y\_coordinate)

{

x = x\_coordinate;

y = y\_coordinate;

}

**4.进程间通信(四种)**

*1、剪贴板*: 剪切板是操作系统维护的一块内存区域，本机所有进程都可以访问;剪切板内存是程序要将数据放入剪切板时分配内存的。只能用于本地进程间通信。延迟提交技术: 使用延迟提交技术时，实际上，直到另一个程序需要数据时，程序才会提供这份数据，也就是，其实我一开始 程序 A 并不往剪贴板中存放真实的数据，如果有其他的 程序 B 访问了剪贴板中的数据，也就是执行了“粘贴”操作，如果剪贴板中存放了数据，那么直接把数据送出去就可以了（这就没有使用延迟提交技术了），而如果剪贴板中没有数据，那么 Windows 就会给上次往剪贴板中存放数据（尽管没有存放实际的数据）的程序，也就是 程序 A发送消息，而后，我们的 程序 A 就可以再次调用 SetClipboardData 来将真实的数据放入到剪贴板中了，就是延迟提交技术

向剪贴板中输入内容：

OpenClipboard(NULL) // 将OpenClipboard 函数的参数指定为 NULL，表明为当前进程打开剪贴板

↓

hGlobalClip = GlobalAlloc(GHND, strlen(pStrData) + 1); //给全局内存对象分配全局内存

↓

pDataBuf = (char \*)GlobalLock(hGlobalClip); //通过给全局内存对象加锁获得对全局内存块的引用

↓

strcpy(pDataBuf, pStrData);

GlobalUnlock(hGlobalClip); //使用完全局内存块后需要对全局内存块解锁

↓

EmptyClipboard(); //清空剪贴板

↓

SetClipboardData(CF\_TEXT, hGlobalClip); // 设置剪贴板数据，这里直接将数据放到了剪贴板中

↓

CloseClipboard() //关闭剪贴板

获取剪贴板内容：

OpenClipboard(NULL) // 打开剪贴板

IsClipboardFormatAvailable(CF\_TEXT) //判断剪贴板中的数据格式是否为 CF\_TEXT

hGlobalClip = GetClipboardData(CF\_TEXT) //从剪贴板中获取格式为 CF\_TEXT 的数据

pDataBuf = (char \*)GlobalLock(hGlobalClip);

GlobalUnlock(hGlobalClip);

*2、匿名管道：*一般用来实现父子进程或兄弟进程之间的通讯，而不能实现跨网络的通信

2.1父进程

CreatePipe(&hRead,&hWrite,&sa,0) // hRead和hWrite分别为 从管道读取数据的句柄 和向管道写入数据的句柄, sa安全描述符，

CreateProcess(\_T("..\\Debug\\Child.exe"),NULL,NULL,NULL,TRUE,0,NULL,NULL,&sui,&pi) //访问指定文件夹中的exe文件，利用此文件来启动子进程

ReadFile(hRead,buf,100,&dwRead,NULL) //从管道读取数据到buf中

WriteFile(hWrite,buf,strlen(buf)+1,&dwWrite,NULL) //从buf中向管道中写入数据

2.2 子进程

ReadFile(hRead,buf,100,&dwRead,NULL) //从管道读取数据到buf中

WriteFile(hWrite,buf,strlen(buf)+1,&dwWrite,NULL) //从buf中向管道中写入数据

void CChildView::OnInitialUpdate() //得到管道的读写句柄

{

CView::OnInitialUpdate();

hRead = GetStdHandle(STD\_INPUT\_HANDLE);

hWrite = GetStdHandle(STD\_OUTPUT\_HANDLE);

}

*3、命名管道*: 通过网络来完成进程间的通信，不同于匿名管道的是命名管道允许无亲缘关系进程间的通信，它可以在不相关的进程之间和不同计算机之间使用。将命名管道作为一种网络编程方案时，它实际上建立了一个C/S通信体系，并在其中可靠的传输数据。与Socket网络通信相比，命名管道不再需要编写身份验证的代码，建立在实际的磁盘介质或文件系统(而不是只存在内存中)

服务器端：

hPipe= CreateNamedPipe("\\\\.\\pipe\\FileMonitor", PIPE\_ACCESS\_DUPLEX | FILE\_FLAG\_OVERLAPPED, 0, 1, 1024, 1024, 0, NULL); //创建命名管道

↓

hEvent = CreateEvent(NULL, TRUE, FALSE, NULL); //创建事件对象

↓

ConnectNamedPipe(hPipe, &ovlap) //等待客户端连接

↓

WaitForSingleObject(hEvent, INFINITE) //一直等待中

↓

CloseHandle(hEvent)

↓

ReadFile(hPipe, cRecvMsg, nBufferLen, &dwLen, NULL); //读取管道中的内容到cRecvMsg中（管道是一种特殊的文件）

WriteFile(hPipe, cSendMsg, strlen(cSendMsg) + 1, &dwLen, NULL); //从cSendMsg中向管道写信息

客户端：

WaitNamedPipe("\\\\.\\pipe\\FileMonitor", NMPWAIT\_WAIT\_FOREVER) //等待连接命名管道

hPipe= CreateFile("\\\\.\\pipe\\FileMonitor", GENERIC\_READ | GENERIC\_WRITE, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL) //打开命名管道

WriteFile(hPipe, cSendMsg, strlen(cSendMsg) + 1, &dwLen, NULL); //从cSendMsg中向管道写信息

ReadFile(hPipe, cRecvMsg, nBufferLen, &dwLen, NULL); //读取管道中的内容到cRecvMsg中

*4、邮槽*（可以跨网络，数据量小）服务器端只能接收数据。客户端只能写入数据CreateMailSlot

**5. MFC消息路由流程**: MFC，Microsoft Foundation Class，微软基础类库，是微软为了简化程序员的开发工作所开发的一套C++类集合，是一套面向对象的函数库，以类的方式提供给用户使用。MFC程序首先定义了一个全局应用程序对象theApp（该对象就表示了应用程序本身，用于把MFC中的类和WinMain函数关联起来），使它在WinMain()函数之前执行，全局对象theApp的构造函数调用时就会先调用其基类CWinApp的构造函数，在基类的构造函数中完成程序的一些初始化的工作，并将应用程序对象的指针保存起来。然后才进入WinMain()函数，MFC中WinMain()函数是程序编译链接时，由链接器将该函数链接到程序中的。WinMain()函数调用AfxWinMain()函数，在AfxWinMain()函数中注册、创建、显示和更新窗口、进入消息循环，因为设置了默认窗口过程故不必再自定义窗口过程。

*1.MFC大致流程：*MFC框架窗口流程：设计和注册窗口(AfxEndDeferRegisterClass)——创建窗口(CreateEx)——显示和更新窗口(ShowWindow(SW\_SHOW)和UpdateWindow())——消息循环(nReturnCode = pThread->Run())

根据基类构造函数this指针，我们的theApp构造完成，在WinMain函数中的指针pThread，pApp就有了值。实际上，WinMain函数功能由AfxWinMain函数完成，AFX表示应用程序框架函数，都是全局函数。

1 重新获取了指向theApp的指针，pThread和pApp所指内容一致。

CWinThread\* pThread = AfxGetThread();

CWinApp\* pApp = AfxGetApp();

2 pApp调用InitApplication()函数，完成MFC内部管理工作。

3 pThread->InitInstance() 完成设计窗口类，注册窗口类，创建窗口，显示窗口，更新窗口

4 nReturnCode = pThread->Run(); 消息循环

*2 MFC消息映射机制*: 消息映射方式：MFC在后台维护了一个句柄和C++对象指针对照表，当收到一个消息后，通过消息结构里资源句柄（查对照表）就可找到与它对应的一个C++对象指针，然后把这个指针传给基类，基类利用这个指针调用WindowProc()函数对消息进行处理，WindowProc()函数中调用OnWndMsg()函数，真正的消息路由及处理是由OnWndMsg()函数完成的。由于WindowProc()和OnWndMsg()都是虚函数，而且是用派生类对象指针调用的，由多态性知最总终调用子类的。在OnWndMsg()函数处理的时候，根据消息种类去查找消息映射，判断所发的消息有没有响应函数，具体方式是到相关的头文件和源文件中寻找消息响应函数声明（从注释宏//{{AFX\_MSG(CDrawView)...//}}AFX\_MSG之间寻找），消息映射（从宏BEGIN\_MESSAGE\_MAP(...)....END\_MESSAGE\_MAP()之间寻找），最终找到对应的消息处理函数;如果是标准消息，则利用消息映射机制查找哪个类响应了当前的这个消息，如果是命令消息，就会交给OnCommand这个函数处理完成消息路由，如果是通知消息，就交给OnNotify处理完成消息路由。后面两个最终会调用OnCmdMsg函数，当然，如果子类中没有对消息进行处理，则消息交由基类处理。

*3 菜单命令消息路由的具体过程：*当点击一个菜单项的时候，最先接受到菜单项消息的是CMainFrame框架类，CMainFrame框架类将会把菜单项消息交给它的子窗口View类，由View类首先进行处理；如果View类检测到没对该菜单项消息做响应，则View类把菜单项消息交由文档类Doc类进行处理；如果Doc类检测到Doc类中也没对该菜单项消息做响应，则Doc类又把该菜单项消息交还给View类，由View类再交还给CMainFrame类处理。如果CMainFrame类查看到CMainFrame类中也没对该消息做响应，则最终交给App类进行处理。处理过程总结：视图view类——文档Doc类——框架CMainFrame类——应用程序Cxxapp类。

*4消息的分类：标准消息，命令消息，通告消息。*[标准消息]：除WM\_COMMAND之外，所有以WM\_开头的消息。[命令消息]：来自菜单、加速键或工具栏按钮的消息。这类消息都以WM\_COMMAND呈现。[通告消息]：由控件产生的消息，例如，按钮的单击，列表框的选择等均产生此类消息，为的是向其父窗口（通常是对话框）通知事件的发生。这类消息也是以WM\_COMMAND形式呈现。