计算机网络

1. TCP连接的三次握手和四次挥手

连接的建立经历以下3个步骤，通常称为“三次握手”，

第一步：客户机向服务器发送一个连接请求报文段，这个特殊的报文段不含应用层数据，其首部的SYN标志位被置为1。另外客户器会随机选择一个起始序号seq=x。

第二步：服务器收到连接请求报文段后，如同意建立连接，就向客户机发回确认报文段，同时为该TCP连接分配缓存和变量！SYN=1，ACK=1，确认号ack=x+1，序号字段seq=y。

第三步：当客户机收到确认报文段后，还要向服务器给出确认，并且也给该连接分配缓存和变量。ACK=1，ack=y+1，seq=x+1。该报文段可以携带数据。

TCP连接的释放过程通常称为“四次挥手”，

第一步：客户机打算关闭连接，就发送一个连接释放报文段，FIN=1，seq=u。TCP是全双工的，当发送FIN报文段后，发送的一方就不能再发数据，但对方还可以发送数据。

第二步：服务器收到该连接释放报文后发送确认报文。ACK=1，ack=u+1，seq=v。此时，TCP连接处于半关闭状态，若服务器发送数据，客户机仍要接收。

第三步:若服务器没有要向客户机发送的数据，那么他就也发一个连接释放报文段。FIN=1，ACK=1，seq=w，ack=u+1.

第四步：客户机收到连接释放报文后发送确认报文。ACK=1，ack=w+1，seq=u+1。（此时TCP连接还没有释放掉，必须等待计时器设置的时间过去后，才进入关闭状态）。

1. 在浏览器中输入[www.baidu.com](http://www.baidu.com)后，执行的过程？

首先，客户端通过DNS解析到该网址对应的IP地址，通过这个IP地址找到客户端到服务器的路径。客户端浏览器发送一个HTTP会话，经过TCP的封装进入网络层。

在客户端的传输层，HTTP会话被分成报文段，添加目的端口和源端口。服务器会利用源端口来向客户端发回相应请求。

客户端的网络层不关心应用层和传输层所传的东西是什么，他只负责选择路径把IP数据报传输到服务器，这里会有一系列的路由协议。

通过链路层

1. 常见协议

DNS协议：用于域名解析服务，将域名地址转换为IP地址。DNS用的端口号是53。FTP21，SMTP25，HTTP80。

NAT协议：网络地址转换，通过将专用网络地址转换为公用地址，从而对外隐藏了内部管理的IP地址。它使得整个专用网只需要一个全球IP地址就可以与因特网连通，大大节省了IP地址消耗。同时，他隐藏了内部网络结构，从而降低了内部网络受到攻击的风险。由于NAT路由器看到了端口，所以他工作在传输层。

ARP协议：地址解析协议，完成IP地址到MAC地址的映射。每个主机都会有一个ARP高速缓存，当目的主机IP地址不在其中时，通过使用目的MAC地址为FF-FF-FF-FF-FF-FF的帧来封装并广播APR请求分组，目的主机收到后，会发出响应分组，主机会把其写入缓存，然后向其发送MAC帧。由于ARP协议看到了IP，所以他工作在网络层。

DHCP：动态主机配置协议，是应用层协议，基于UDP！需要IP的主机广播发送发现报文，DHCP服务器会做出响应：广播“DHCP提供”消息，询问是否接受自己提供的IP。客户机接到，如果接受就发送请求，然后服务器把IP提供给客户机。

ICMP：网际控制报文协议，属于网络层协议！ICMP报文有ICMP差错报告报文和ICMP询问报文。以下五种情况会发送差错报文：

（1）路由器或主机不能交付数据报，终点不可达。（2）路由器或主机因拥塞而丢弃数据报，源点抑制。（3）数据报TTL为0，时间超过。（4）数据报首部有字段错误时丢弃，发回参数问题。（5）有更好路径，发回路由重定向。以下情况不发ICMP：对ICMP差错报文不再发送ICMP。对第一个分片的后续数据报片不再发送。对组播数据报不发送。对特殊地址的数据报不发送。

PING应用了ICMP查询报文（有四种类型）。

路由选择协议：RIP路由协议，是应用层协议，用UDP。OSPF路由协议，是网络层协议，直接用IP数据报传送。

操作系统

1. 进程与线程的区别？

定义，进程：一个程序在一个数据集合上的一次运行过程。线程：进程中的一个实体，是被系统独立调度和执行的基本单位。

1）系统开销。由于创建或撤销进程时，系统要为之分配或回收资源，如内存空间、I/O设备等，所以操作系统付出的开销远大于创建或撤销线程时的开销。类似的，进程间切换时，涉及 CPU环境的保存及设置，而线程只需保存和设置少量寄存器内容，开销很小。此外，线程共享进程的地址空间，故同步与通信非常容易实现。

2）进程之间地址空间相互独立，同一进程间线程共享进程资源。

3）通信方面，进程间通信（IPC）需要进程同步和互斥手段的辅助，而线程间可以直接读写进程数据段（如全局变量）来通信。

4）拥有资源。进程时拥有资源的基本单元，而线程不拥有系统资源 ，这也是切换线程开销小的一个原因。但线程可以访问其隶属进程的系统资源。

5）调度。线程是独立调度的基本单位。在同一进程中线程的切换不会引起进程切换，但不同进程间进行线程切换，会引起进程切换。

1. 进程间的通信

1）共享内存。用户进程空间是独立的，进程运行期间不能访问其他进程的空间。但进程间可以存在一块可直接访问的共享空间，通过对这片共享空间进行读写实现进程之间的信息交换。

2）消息队列。在消息传递系统中，进程间的数据交换是以格式化的消息为单位的。

3）管道。管道是消息传递的一种特殊方式。所谓管道，是指用于连接一个读进程和一个写进程以实现它们之间通信的一个共享文件。

1. 什么是死锁？产生的原因？处理策略？

死锁是指多个进程因竞争资源而造成的一种僵局，若无外力作用，这些进程都将无法向前推进。

死锁产生必须同时满足以下四个条件：

1）互斥条件：在一段时间内某资源仅为一个进程占有，此时若有其他进程请求该资源，则请求进程只能等待。

2）不剥夺条件：进程所获得的资源在未使用完毕前，不能被其他进程强行夺走。

3）请求和保持：进程已经保持了至少一个资源，但又提出了新的资源请求，而该资源已被其他进程占有。此时请求进程被阻塞，但对自己已获得的资源保持不放。

4）循环等待条件：存在一种进程资源的循环等待链。资源分配图含圈但系统不一定有死锁，因为同类资源数可能大于1。但若系统中每类资源只有一个，则资源分配图含圈就是出现死锁的重要条件了。

死锁的处理策略又可分为三种：预防、避免、检测及解除。

预防：破坏上述四个条件中的一种，效率很低。

避免：需要知道将来的资源需求，然后判断是否可能死锁。安全状态是指系统能按某种推进顺序，为每个进程分配其所需的资源。系统处于安全状态一定不会进入死锁，处于不安全状态则有可能进入死锁状态。银行家算法是最著名的死锁避免算法。比较Available的资源和每个进程需求的资源（都是向量），需求的资源若小于Available，则分给他后再回收资源，然后再对剩下的进程执行相同过程。

检测和解除：资源分配图，方框匡着圆点，一个方框是一类资源，圆点个数代表资源个数，用圆圈代表进程。方框指向圆圈的线表示进程得到了某类资源的一个，获得多个需用多条线；圆圈指向方框表示进程需要一个某类资源。某类资源空闲数就是框中资源个数减去出度，若某个进程所有请求都小于等于资源空闲数，那么就可以消去他所有请求边和分配边。S为死锁的条件是当且仅当s状态的资源分配图是不可完全简化的，该条件称为死锁定理。

1. Windows内存管理的几种方式和优缺点？

windows 内存管理方式主要分为：页式管理，段式管理，段页式管理。

页式管理：将各进程的虚拟空间划分为若干个长度相等的页；页式管理把内存空间按照页的大小划分成片或者页面，然后把页式虚拟地址与内存地址建立一一对应的页表；并用相应的硬件地址变换机构来解决离散地址变换问题。其优点是没有外碎片，每个内碎片不超过页的大小。缺点是，程序全部装入内存，要求有相应的硬件支持。例如地址变换机构缺页中断的产生和选择淘汰页面等都要求有相应的硬件支持。这增加了机器成本，增加了系统开销。

段式管理：把程序按照内容或过程函数关系分段，每段都有自己的名字。段式管理程序以段为单位分配内存，然后通过地址映射机构把段式虚拟地址转换为实际内存物理地址。其优点是可以分别编写和编译，可以针对不同类型的段采用不同的保护，可以按段为单位来进行共享，包括通过动态链接进行代码共享。缺点是会产生碎片。

段页式管理：为了实现段页式管理，系统必须为每个作业或进程建立一张段表以管理内存分配与释放、缺段处理等。另外由于一个段又被划分成了若干个页。每个段必须建立一张页表以把段中的虚页变换成内存中的实际页面。显然与页式管理时相同，页表中也要有相应的实现缺页中断处理和页面保护等功能的表项。段页式管理的段式管理与页式管理方案结合而成的所以具有他们两者的优点。但反过来说，由于管理软件的增加，复杂性和开销也就随之增加了。另外需要的硬件以及占用的内存也有所增加。使得速度降下来。

编程语言

1. Static关键字

Static有两种用法：面向过程的static和面向对象的static。

面向过程：

1）静态全局变量：在全局变量前加上static，该变量就被定义成为一个静态全局变量。一，该变量在全局数据区分配内存。二，未经初始化的静态全局变量会被自动初始化为0。三，静态全局变量在它的整个文件都是可见的，在文件之外是不可见的。

2）静态局部变量：在局部变量前加上static，该变量就被定义称为一个静态局部变量。一，该变量在全局数据区分配内存。二，静态局部变量在程序执行到该对象时首次初始化，但下次再执行时不再初始化！即保留之前的值。三，始终驻留在全局数据区，直到程序运行结束，但他的作用域是局部作用域，仅限于定义他的函数内。

3）静态函数：静态函数仅在定义它的文件可见，不能被其他文件所用。

面向对象：

1）静态数据成员：一，对于非静态数据成员，每个类对象都有自己的拷贝。而对于静态数据成员，无论这个类的对象被定义了多少个，静态数据成员在程序中也只有一份拷贝，由该类型的所有对象共享访问。也就是说。因为它不属于特定的类对象，所以在没有产生类对象时其作用域就可见，即在没有产生类的实例时，我们就可以操作它。二，静态数据成员存储在全局数据区，在定义它的时候要为他分配空间，所以它不能在类声明中定义。三，静态数据成员与普通数据成员一样遵从public、protected、private访问规则。

2）静态成员函数：普通的成员函数一般都隐含了一个this指针，this指针指向类的对象本身，因为普通成员函数总是具体的属于某个类的具体对象的。通常情况下，this是缺省的。如函数fn()实际上是this->fn()。但是与普通函数相比，静态成员函数由于不是与任何的对象相联系，因此它不具有this指针。从这个意义上讲，它无法访问属于类对象的非静态数据成员，也无法访问非静态成员函数，它只能调用其余的静态成员函数。

1. 分别介绍const和#define，然后说明二者的区别。

const用于定义变量为常类型，一般有以下几个作用：1）定义const常量，具有不可变性，增强了程序的健壮性。2）进行类型检查。3）节省空间，避免不必要的内存分配。

const与define两者都可以用来定义常量，但是const定义时，定义了常量的类型，所以更精确一些。#define只是简单的文本替换。const与指针和引用！const的几种使用：用于指针的两种情况，限定函数的传递值参数，限定函数返回值型，限定函数类型（const成员函数不改变对象的成员函数）。

为什么使用const？我们定义的类的成员函数中，常常有一些成员函数不改变类的数据成员，也就是说，这些函数是"只读"函数，而有一些函数要修改类数据成员的值。如果把不改变数据成员的函数都加上const关键字进行标识，显然，可提高程序的可读性。其实，它还能提高程序的可靠性，已定义成const的成员函数，一旦企图修改数据成员的值，则编译器按错误处理。

1. new/delete,malloc/free的区别？

它们的相同点在于：c++中，申请动态内存与释放动态内存时，用两者都可以，而且它们存储方式相同，都位于堆中。不同点在于：malloc/free是标准库函数，在c语言中位于<stdlib.h>头文件，而new/delete是C++的运算符。对类对象而言，malloc/free不能满足动态对象的要求，动态对象的要求：对象在创建时时自动执行构造函数，消亡前自动执行析构函数，这是因为malloc/free不在编译器控制范围之内！相应于以上几点，它们的区别可以具体总结如下：

1. new能自动计算需要分配的内存空间，而malloc需手工计算字节数。
2. new直接带具体类型的指针，malloc返回void类型的指针。
3. 上述特点决定了new是类型安全的，malloc不是。
4. new将调用构造函数，malloc不能；delete调用析构函数，free不能。
5. malloc/free需库文件支持，new/delete不用。
6. 什么是内存泄露？

所谓内存泄露是指由于疏忽或错误造成程序未能释放已经不再使用的内存的情况，一般指的是堆内存的泄露。并非是说内存在物理上的消失，而是应用程序分配某段内存后，由于涉及错误，失去了对该段内存的控制，因为造成了内存的浪费。

1. 各种数据类型占内存大小？struct和union的区别？类所占内存大小的计算？

32位系统，char：1字节，char\*：4字节，short int：2字节，int：4字节，float：4字节，double：8字节，long：4字节，long long：8字节，unsigned long：4字节。

64位系统仅有三个发生变化，char\*：8字节，long：8字节，unsigned long：8字节。

1）struct和union都是由多个不同的数据类型成员组成, 但在任何同一时刻, union中只存放了一个被选中的成员， 而struct的所有成员都存在。2）在struct中，各成员都占有自己的内存空间，它们是同时存在的,一个struct变量的总长度等于所有成员长度之和，遵从字节对齐原则; 在Union中，所有成员不能同时占用它的内存空间，它们不能同时存在 , Union变量的长度等于最长的成员的长度（注意！union大小为内部变量的最大大小，然后按照最大类型的大小进行调整）。3）对于union的不同成员赋值, 将会对其它成员重写, 原来成员的值就不存在了,所以，共同体变量中起作用的成员是最后一次存放的成员; 而对于struct的不同成员赋值是互不影响的。

空的类也是会占用内存空间的，而且大小是1，原因是C++要求每个实例在内存中都有独一无二的地址。

（一）类内部的成员变量：

普通的变量：是要占用内存的，但是要注意内存对齐（这点和struct类型很相似）。

static修饰的静态变量：不占用内存，原因是编译器将其放在全局变量区。

从父类继承的变量：计算进子类中

（二）类内部的成员函数：

非虚函数(构造函数、静态函数、成员函数等)：不占用内存。

虚函数：要占用4个字节(32位的操作系统)，用来指定虚拟函数表的入口地址。跟虚函数的个数没有关系。父类子类工享一个虚函数指针。

构成对象本身的只有数据，任何成员函数都不隶属于任何一个对象，非静态成员函数与对象的关系就是绑定，绑定的中介就是this指针。

1. 类成员函数的重载、覆盖的区别？

成员函数被重载的特征：1）相同的范围（在同一个类中）；2）函数名字相同；3）参数不同；4）virtual 关键字可有可无。

覆盖是指派生类函数覆盖基类函数，特征是：1）不同的范围（分别位于派生类与基类）；2）函数名字相同；3）参数相同；4）基类函数必须有virtual 关键字。

1. 哈希函数及冲突处理方法？

哈希函数有平方取中法，伪随机数法，除留余数法，最后一种是最常用的。

冲突处理有以下三种常见方法：

1）分离链接法。也称拉链法，将所有关键字为同义词的结点链接在同一个单链表中。缺点是指针需要额外的空间。

2）开放定址法。冲突后选择另外的单元，将原地址加上某个数后再取余：Hi=（H（key）+di）% m i=1，2，…，n，其中H（k）就是首次哈希时冲突的单元。di的选择有以下几种：1线性探测法di=i。2平方探测法di=i\*i。

3）再哈希法。这种方法是同时构造多个不同的哈希函数。