概念问题

C++/数据结构

1、 简述你对“面向对象”和“面向过程”编程思想的认识与思考

例：考虑一个工资系统

- 这种功能明确的系统确实适合结构化的开发方法：自顶向下，逐步求精。
- 你完美地完成了任务。

- 而后，极有可能，客户会跑过来跟你说：
  - 给我加个统计报表撒
  - 下个月我想一些人发记时工资，一些人发计件工资啊行啊，有人每月一发，有人每周一发哦
  - 加个个人所得税系统接口
  - 加个图像用户界面，用交互查询

2013/9/18 Institute of Computer Software Nanjing University 9

面向过程就是分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候一个一个依次调用就可以了。

面向对象是把构成问题事务分解成各个对象，建立对象的目的不是为了完成一个步骤，而是为了描叙某个事物在整个解决问题的步骤中的行为。

例如五子棋，面向过程的设计思路就是首先分析问题的步骤：1、开始游戏，2、黑子先走，3、绘制画面，4、判断输赢，5、轮到白子，6、绘制画面，7、判断输赢，8、返回步骤 2，9、输出最后结果。把上面每个步骤用分别的函数来实现，问题就解决了。

而面向对象的设计则是从另外的思路来解决问题。整个五子棋可以分为 1、黑白双方，这两方的行为是一模一样的，2、棋盘系统，负责绘制画面，3、规则系统，负责判定诸如犯规、输赢等。第一类对象（玩家对象）负责接受用户输入，并告知第二类对象（棋盘对象）棋子布局的变化，棋盘对象接收到了棋子的 i 变化就要负责在屏幕上面显示出这种变化，同时利用第三类对象（规则系统）来对棋局进行判定。

可以明显地看出，面向对象是以功能来划分问题，而不是步骤。同样是绘制棋局，这样的行为在面向过程的设计中分散在了总多步骤中，很可能出现不同的绘制版本，因为通常设计人员会考虑到实际情况进行各种各样的简化。而面向对象的设计中，绘图只可能在棋盘对象中出现，从而保证了绘图的统一。

功能上的统一保证了面向对象设计的可扩展性。比如我要加入悔棋的功能，如果要改动面向过程的设计，那么从输入到判断到显示这一连串的步骤都要改动，甚至步骤之间的循序都要进行大规模调整。如果是面向对象的话，只用改动棋盘对象就行了，棋盘系统保存了黑白双方的棋谱，简单回溯就可以了，而显示和规则判断则不用顾及，同时整个对对象功能的调用顺序都没有变化，改动只是局部的。

再比如我要把这个五子棋游戏改为围棋游戏，如果你是面向过程设计，那么五子棋的规则就分布在了你的程序的每一个角落，要改动还不如重写。但是如果你当初就是面向对象的设计，那么你只用改动规则对象就可以了，五子棋和围棋的区别不就是规则吗？（当然棋盘大小好像也不一样，但是你会觉得这是一个难题吗？直接在棋盘对象中进行一番小改动就可以了。）而下棋的大致步骤从面向对象的角度来看没有任何变化。

当然，要达到改动只是局部的需要设计的人有足够的经验，使用对象不能保证你的程序就是面向对象，初学者或者很蹩脚的程序员很可能以面向对象之虚而行面向过程之实，这样设计出来的所谓面向对象的程序很难有良好的可移植性和可扩展性。

2、 ADT 是什么？简述你对“数据抽象”和“信息隐藏”的认识

抽象数据类型(Abstract Data Type 简称 ADT)是指一个数学模型以及定义在此数学模型上的一组操作。抽象数据类型需要通过固有数据类型（高级编程语言中已实现的数据类型）来实现。**抽象数据类型是与表示无关的数据类型**，是一个数据模型及定义在该模型上的一组运算。对一个抽象数据类型进行定义时，必须给出它的名字及各运算的运算符名，即函数名，并且规定这些函数的参数性质。一旦定义了一个抽象数据类型及具体实现，程序设计中就可以像使用基本数据类型那样，十分方便地使用抽象数据类型。

抽象数据类型通过类(class)实现

- 程序设计语言对抽象数据类型的支持是指允许用户自定义具有如下特征的数据类型：
    1. 模块封装：The representation of, and operations on, objects of the type are defined in a single syntactic unit
    2. 信息隐蔽：The representation of objects of the type is hidden from the program units that use these objects, so the only operations possible are those provided in the type's definition

3、const 和 static 有什么作用？

`const` 是一个 C 和 C++语言的关键字，它限定一个变量不允许被改变，即只读。使用 `const` 在一定程度上可以提高程序的安全性和可靠性，也便于实现对此进行优化（如把只读对象放入 ROM 中）。`const` 作为类型限定符，是类型的一部分。

静态变量（*Static Variable*）在计算机编程领域指在程序执行前系统就为之静态分配（也即在运行时中不再改变分配情况）存储空间的一类变量。与之相对应的是在运行时只暂时存在的自动变量（即局部变量）与以动态分配方式获取存储空间的一些对象，其中自动变量的存储空间在调用栈上分配与释放。

初始化为非零值的静态分配数据和全局数据存在数据段中，运行相同程序的每个进程都有自己的数据段。
初始化为零值的静态分配数据和全局数据存放在进程的 BSS 区域内。
BSS 段： BSS 段（bss segment）通常是指用来存放程序中未初始化的全局变量的一块内存区域。BSS 是英文 Block Started by Symbol 的简称。BSS 段属于静态内存分配。

是“Block Started bySymbol”的缩写，意为“以符号开始的块”。

BSS 是 Unix 链接器产生的未初始化数据段。其他的段分别是包含程序代码的"text"段和包含已初始化数据的"data"段。BSS 段的变量只有名称和大小却没有值。此名后来被许多文件格式使用，包括 PE。"以符号开始的块"指的是编译器处理未初始化数据的地方。BSS 节不包含任何数据，只是简单的维护开始和结束的地址，以便内存区能在运行时被有效地清零。BSS 节在应用程序的二进制映象文件中并不存在。

数据段： 数据段（data segment）通常是指用来存放程序中已初始化的全局变量的一块内存区域。数据段属于静态内存分配。**静态变量**存放在 data 段中

代码段： 代码段（code segment/text segment）通常是指用来存放程序执行代码的一块内存区域。这部分区域的大小在程序运行前就已经确定，并且内存区域通常属于只读，某些架构也允许代码段为可写，即允许修改程序。在代码段中，也有可能包含一些只读的常数变量，例如字符串常量等。**代码段是存放了程序代码的数据，假如机器中有数个进程运行相同的一个程序，那么它们就可以使用同一个代码段。**

堆（heap）：堆是用于存放进程运行中被**动态分配的内存段**，它的大小并不固定，可动态扩张或缩减。当进程调用 malloc 等函数分配内存时，新分配的内存就被动态添加到堆上（堆被扩张）；当利用 free 等函数释放内存时，被释放的内存从堆中被剔除（堆被缩减）

栈(stack)：栈又称堆栈， 是用户存放程序临时创建的**局部变量**，也就是说我们函数括弧"{}"中定义的变量（但不包括 static 声明的变量，static 意味着在数据段中存放变量）。除此以外，在函数被调用时，其参数也会被压入发起调用的进程栈中，并且待到调用结束后，函数的返回值也会被存放回栈中。由于栈的先进先出特点，所以栈特别方便用来保存/恢复调用现场。从这个意义上讲，我们可以把堆栈看成一个寄存、交换临时数据的内存区。

4、友元关系的利与弊

如果将一个函数或一个类声明为另一个类的友元，那么它就可以直接存取这个类对象中的各种数据，而不必在意这些数据的封装级别，即无论是 private 的，protected 的，还是 public 的，有钱同使，有难同当。

关于友元利弊的分析： **面向对象程序设计的一个基本原则是封装性和信息隐蔽，而友元却可以访问其他类中的私有成员，不能不说这是对封装原则的一个小的破坏**。但是它能有助于数据共享，能提高程序的效率，在使用友元时，要注意到它的副作用，不要过多地使用友元，只有在使用它能使程序精炼，并能大大提高程序的效率时才用友元。

为什么有些操作符重载(<<，>>)必须要用友元函数而不能用成员函数？

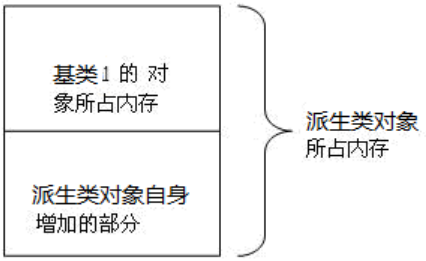
如果是重载双目操作符（即为类的成员函数），就只要设置一个参数作为右侧运算量，而左侧运算量就是对象本身。。。。。

而 >> 或<< 左侧运算量是 cin 或 cout 而不是对象本身，所以不满足后面一点。。。。。。。就只能申明为友元函数了。。。

5、C++多态的实现

1. 用 virtual 关键字申明的函数叫做虚函数，虚函数肯定是类的成员函数。
2. 存在虚函数的类都有一个一维的虚函数表叫做虚表。类的对象有一个指向虚表开始的虚指针。虚表是和类对应的，虚表指针是和对象对应的。
3. 多态性是一个接口多种实现，是面向对象的核心。分为类的多态性和函数的多态性。
4. 多态用虚函数来实现，结合动态绑定。
5. 纯虚函数是虚函数再加上= 0。
6. 抽象类是指包括至少一个纯虚函数的类。

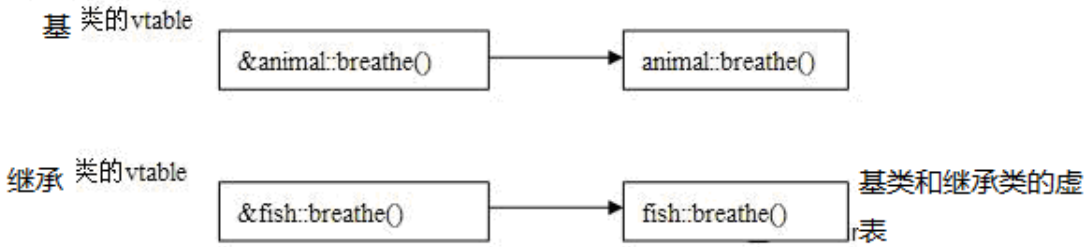
http://blog.csdn.net/tujiaw/article/details/6753498



构造函数顺序：
基类构造函数→派生类构造函数

前面输出的结果是因为编译器在编译的时候，就已经确定了对象调用的函数的地址，要解决这个问题就要使用**迟绑定（late binding）技术**。当编译器使用迟绑定时，就会在运行时再去确定对象的类型以及正确的调用函数。而要让编译器采用迟绑定，就要在基类中声明函数时使用 virtual 关键字（注意，这是必须的，很多学员就是因为没有使用虚函数而写出很多错误的例子），这样的函数我们称为虚函数。一旦某个函数在基类中声明为 virtual，那么在所有的派生类中该函数都是 virtual，而不需要再显式地声明为 virtual。
前面输出的结果是因为编译器在编译的时候，就已经确定了对象调用的函数的地址，要解决这个问题就要使用迟绑定（late binding）技术。当编译器使用迟绑定时，就会在运行时再去确定对象的类型以及正确的调用函数。而要让编译器采用迟绑定，就要在基类中声明函数时使用 virtual 关键字（注意，这是必须的，很多学员就是因为没有使用虚函数而写出很多错误的例子），这样的函数我们称为虚函数。一旦某个函数在基类中声明为 virtual，那么在所有的派生类中该函数都是 virtual，而不需要再显式地声明为 virtual。

编译器在编译的时候，发现基类中有虚函数，此时编译器会为每个包含虚函数的类创建一个虚表（即 vtable），该表是一个一维数组，在这个数组中存放每个虚函数的地址。



那么如何定位虚表呢？编译器另外还为每个类的对象提供了一个虚表指针（即 vptr），这个指针指向了对象所属类的虚表。在程序运行时，根据对象的类型去初始化 vptr，从而让 vptr 正确的指向所属类的虚表，从而在调用虚函数时，就能够找到正确的函数。对于例 1-2 的程序，由于 pAn 实际指向的对象类型是 fish，因此 vptr 指向的 fish 类的 vtable，当调用 pAn->breathe()时，根据虚表中的函数地址找到的就是 fish 类的 breathe()函数。

那么虚表指针在什么时候，或者说在什么地方初始化呢？

答案是在构造函数中进行虚表的创建和虚表指针的初始化。还记得构造函数的调用顺序吗，在构造子类对象时，要先调用父类的构造函数，此时编译器只“看到了”父类，并不知道后面是否后还有继承者，它初始化父类对象的虚表指针，该虚表指针指向父类的虚表。当执行子类的构造函数时，子类对象的虚表指针被初始化，指向自身的虚表。对于例 2-2 的程序来说，当 fish 类的 fh 对象构造完毕后，其内部的虚表指针也就被初始化为指向 fish 类的虚表。在类型转换后，调用 pAn->breathe()，由于 pAn 实际指向的是 fish 类的对象，该对象内部的虚表指针指向的是 fish 类的虚表，因此最终调用的是 fish 类的 breathe()函数。
要注意：对于虚函数调用来说，每一个对象内部都有一个虚表指针，该虚表指针被初始化为本类的虚表。所以在程序中，不管你的对象类型如何转换，但该对象内部的虚表指针是固定的，所以呢，才能实现动态的对象函数调用，这就是 C++多态性实现的原理。

6、STL 是什么？组成部分和核心作用

标准模板库（英文：**Standard Template Library**，缩写：**STL**），是一个 C++软件库，也是 C++标准程序库的一部分。其中包含 5 个组件，分别为算法、容器、迭代器、函数、适配器。容器即物之所属；算法是解决问题的方式；迭代器是对容器的访问逻辑的抽象，是连接算法和容器的纽带，通过添加了一种间接层的方式实现了容器和算法之间的独立。本文从应用的角度对 STL 的方方面面进行了简单的介绍。

模板是 C++程序设计语言中的一个重要特征，而标准模板库正是基于此特征。标准模板库使得 C++编程语言在有了同 Java 一样强大的类库的同时，保有了更大的可扩展性。

标准模板库于 1994 年 2 月年正式成为 ANSI/ISO C++的一部份，它的出现，促使 C++程序员的思维方式更朝向**泛型编程（generic program）**发展。

7、阐述C++在什么情况下必须进行运算符重载。

？

8、 为什么说“继承是C++面向对象的一个主要特征之一”，请做一下简要说明。

？

9、 请说明函数模板(Function Template)和函数模板实例化(function-template specification)的区别和联系。

函数模板实例化

在函数模板为每个类型时首先调用中，编译器创建一个实例化。 每个实例化是为该类型的该模板化功能的版本。 在中，此函数为类型时，使用此实例化将调用。 如果您有几个相同的实例化，即使在不同的模块，因此，只有该实例化的一个副本在可执行文件将结果。

函数参数将所有参数的函数模板允许和参数，对该参数不依赖于模板参数的位置。

函数模板可以通过声明与特定类型的模板显式实例化作为参数。

C++中提供了函数模板，实际上是建立一个通用函数，其函数类型和形参类型不具体指定，用一个虚拟的类型来代表，这个通用函数就成为函数模板。使用模板的好处就是对于那些函数体相同的函数都可以用这个模板来代替，而不必去定义每个具体的函数去实现。下面通过一个简单的具体例子（比较两个数的大小）来说明：

```
#include <iostream>
using namespace std;

template<class T>     //模板声明，T 为类型参数
T Max(T a, T b)       //定义一个通用函数，用 T 作虚拟的类型名
{
 if (a>b)
 {
  return a;
 }
 else
  return b;
}
```

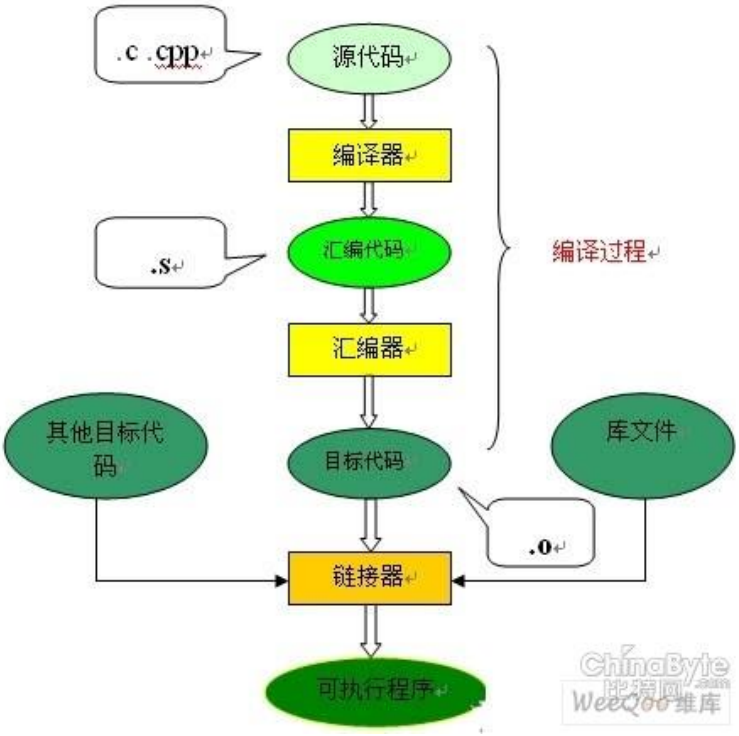
模板（**Template**）指 C++程序设计语言中的函数模板与类别模板，大体对应于 java 和 C＃中的泛型。目前，模板已经成为 C++的泛型编程中不可缺少的一部分。

模板定义以关键字 template 开始，后接模板形参表，模板形参表是用尖括号括住的一个或者多个模板形参的列表，形参之间以逗号分隔。 模板形参可以是表示类型的类型形参，也可以是表示常量表达式的非类型形参。非类型形参跟在类型说明符之后声明。类型形参跟在关键字 class 或 typename 之后定义。

模板是 C++程序员绝佳的武器，特别是结合了多重继承（multiple inheritance）与运算符重载（operator overloading）之后。C++ 的标准库提供许多有用的函数大多结合了模板的观念，如 STL 以及 IO Stream。

模板实例化（template instantiation ）是指在编译或链接时生成函数模板或类模板的具体实例源代码。ISO C++定义了两种模板实例化方法：隐式实例化（当使用实例化的模板时自动地在当前代码单元之前插入模板的实例化代码）、显式实例化（直接声明模板实例化）。

10、编译和链接的过程



源文件的编译过程包含两个主要阶段，而它们之间的转换是自动的。第一个阶段是预处理阶段，在正式的编译阶段之前进行。预处理阶段将根据已放置在文件中的预处理指令来修改源文件的内容。#include 指令就是一个预处理指令，它把头文件的内容添加到.cpp 文件中还有其他许多预处理指令

这个在编译之前修改源文件的方式提供了很大的灵活性，以适应不同的计算机和操作系统环境的限制。一个环境需要的代码跟另一个环境所需的代码可能有所不同，因为可用的硬件或操作系统是不同的。在许多情况下，可以把用于不同环境的代码放在同一个文件中，再在预处理阶段修改代码，使之适应当前的环境。

预处理器显示为一个独立的操作，但一般不能独立于编译器来执行这个操作。调用编译器会自动执行预处理过程，之后才编译代码。

编译器为给定源文件输出的是机器码，执行这个过程需要较长时间。在对象文件之间并没有建立任何连接。对应于某个源文件的对象文件包含在其他源文件中定义的函数引用或其他指定项的引用，而这些函数或项仍没有被解析。同样，也没有建立同库函数的链接。实际上，这些函数的代码并不是文件的一部分。这些工作是由链接程序(有时称为链接编辑器)完成的

链接程序把所有对象文件中的机器码组合在一起，并解析它们之间的交叉引用。它还集成了对象模块所使用的库函数的代码。这是链接程序的一种简化表示，因为这里假定在可执行模块中，模块之间的所有链接都是静态建立

的。实际上有些链接是动态的，即这些链接是在程序执行时建立的。

链接程序静态地建立函数之间的链接，即在程序执行之前建立组成程序的源文件中所包含的函数链接。动态建立的函数之间的链接(在程序执行过程中建立的链接)将函数编译并链接起来，创建另一种可执行模块—— 动态链接库或共享库。动态链接库中的函数链接是在程序调用函数时才建立的，在程序调用之前，该链接是不存在的。

动态链接库有几个重要的优点。一个主要的优点是动态链接库中的函数可以在几个并行执行的程序之间共享，这将节省相同函数占用的内存空间。另一个优点是动态链接库在调用其中的函数之前是不会加载到内存中的。也就是说，如果不使用给定动态链接库中的函数，该动态链接库就不会占用内存空间

11、解释“优先级队列”这一抽象数据类型及实现方法

如果我们给每个元素都分配一个数字来标记其优先级，不妨设较小的数字具有较高的优先级，这样我们就可以在一个集合中访问优先级最高的元素并对其进行查找和删除操作了。这样，我们就引入了优先级队列 这种数据结构。

缺省情况下，优先级队列利用一个最大堆完成

函数列表：

empty() 如果优先队列为空，则返回真

pop() 删除第一个元素

push() 加入一个元素

size() 返回优先队列中拥有的元素的个数

top() 返回优先队列中有最高优先级的元素

用途就不用多说了吧，例如 Huffman 编码、分支限界、A*启发式都需要用到优先队列存放信息。

12、逆波兰式用什么数据结构算法的效率比较高，为什么

逆波兰式就是**后缀表达式**！用栈结构来搞算法效率比较高。

下面以(a+b)*c 为例子进行说明：

(a+b)*c 的逆波兰式为 ab+c*，假设计算机把 ab+c*按从左到右的顺序压入栈中，并且按照遇到运算符就把栈顶两个元素出栈，执行运算，得到的结果再入栈的原则来进行处理，那么 ab+c*的执行结果如下：

1）a 入栈（0 位置）

2）b 入栈（1 位置）

3）遇到运算符“+”，将 a 和 b 出栈，执行 a+b 的操作，得到结果 d=a+b，再将 d 入栈（0 位置）

4）c 入栈（1 位置）

5）遇到运算符“*”，将 d 和 c 出栈，执行 d*c 的操作，得到结果 e，再将 e 入栈（0 位置）

经过以上运算，计算机就可以得到(a+b)*c 的运算结果 e 了。

逆波兰式除了可以实现上述类型的运算，它还可以派生出许多新的算法，数据结构，这就需要灵活运用了。逆波兰式只是一种序列体现形式。

13、C 和 C++，C++和 Java 的区别

C 是一个结构化语言，如谭老爷子所说：它的重点在于算法和数据结构。C 程序的设计首要考虑的是如何通过一个过程，对输入（或环境条件）进行运算处理得到输出（或实现过程（事务）控制），而对于 C++，首要考虑的是如何构造一个对象模型，让这个模型能够契合与之对应的问题域，这样就可以通过获取对象的状态信息得到输出或实现过程（事务）控制。

所以 C 与 C++的最大区别在于它们的用于解决问题的思想方法不一样。之所以说 C++比 C 更先进，是因为面向对

象 设计这个概念已经被融入到 C++之中 ”，而就语言本身而言，在 C 中更多的是算法的概念。那么是不是 C 就不重要了，错！算法是程序设计的基础，好的设计如果没有好的算法，一样不行。而且，“C 加上好的设计”也能写出非常好的东西。

对语言本身而言，C 是 C++的子集，那么是什么样的一个子集？从上文可以看出， C 实现了 C++中过程化控制及其它相关功能，而在 C++中的 C（我称它为“C+”），相对于原来的 C 还有所加强，引入了重载、内联函数、异常处理等等玩艺儿，C++更是拓展了面向对象设计的内容，如类、继承、虚函数、模板和包容器类等等。
再提高一点，在 C++中，数据封装、类型这些东东已不是什么新鲜事了，需要考虑的是诸如：对象粒度的选择、对象接口的设计和继承、组合与继承的使用等等问题。
所以相对于 C，C++包含了更丰富的“设计”的概念，但 C 是 C++的一个自洽子集，也具有强大的功能，同样值得学习。

	C++	**Java**
指针	有	JAVA 语言让编程者无法找到指针来直接访问内存无指针，并且增添了自动的内存管理功能，从而有效地防止了 c / c++语言中指针操作失误，如野指针所造成的系统崩溃
多重继承	c++支持多重继承，这是 c++的一个特征，它允许多父类派生一个类。尽管多重继承功能很强，但使用复杂，而且会引起许多麻烦，编译程序实现它也很不容易。 多重继承的问题，二义性：两个基类 A 都有同一个 virtual 方法，一个子类继承了两个基类，并同时 override 了这个方法，那该听谁的？	Java 不支持多重继承，但允许一个类继承多个接口(extends+implement)，实现了 c++多重继承的功能，又避免了 c++中的多重继承实现方式带来的诸多不便。 解决多重继承的问题：接口中没有实现，所以也就没有刚才的问题了。
全局变量 结构体 struct	允许	不允许
自动内存管理	new 完了要显式 delete 掉	内存垃圾回收
操作符重载	支持	不支持
缺省函数参数（int a = 1）	支持	不支持（不过可以通过 override 实现）
goto 语句	支持（不提倡）	不支持（但是保留关键字）
类型转换	支持隐式转换	不支持隐式转换

14、什么是预处理

程序设计中的预处理(Preprocess)，程序设计领域，预处理是在程序源代码被编译之前，由预处理器（Preprocessor）对程序源代码进行的处理。这个过程并不对程序的源代码进行解析，但它把源代码分割或处理成为特定的符号用来支持宏调用。

预处理器的主要作用就是把通过预处理的内建功能对一个资源进行等价替换，最常见的预处理有：文件包含，条件编译、布局控制和宏替换 4 种。

15、堆和栈的区别

栈区（stack）— 由编译器自动分配释放 ，存放函数的参数值，局部变量的值等。其操作方式类似于数据结构中的栈。

堆区（heap） — 一般由程序员分配释放， 若程序员不释放，程序结束时可能由 OS 回收 。注意它与数据结构中的堆是两回事，分配方式倒是类似于链表，呵呵。

堆（英语：**heap**)亦被称为：**优先队列**（英语：**priority queue**），是计算机科学中一类特殊的数据结构的统称。堆通常是一个可以被看做一棵树的数组对象。在队列中，调度程序反复提取队列中第一个作业并运行，因而实际情况中某些时间较短的任务将等待很长时间才能结束，或者某些不短小，但具有重要性的作业，同样应当具有优先权。堆即为解决此类问题设计的一种数据结构。

堆栈（英文：**stack**），也可直接称**栈**。台湾作**堆疊**，在计算机科学中，是一种特殊的串行形式的数据结构，它的特殊之处在于只能允许在链结串行或阵列的一端（称为堆栈顶端指标，英文为 top）进行加入资料（push）和输出资料（pop）的运算。另外堆栈也可以用一维阵列或连结串行的形式来完成。堆栈的另外一个相对的操作方式称为伫列。

由于堆栈数据结构只允许在一端进行操作，因而按照后进先出（LIFO, Last In First Out）的原理运作。

堆栈数据结构使用两种基本操作：推入（push）和弹出（pop）：

16、C, C++分别如何处理调用次数特别高的函数

宏（**Macro**），是一种批量批量处理的称谓。 计算机科学里的宏是一种抽象(Abstraction)，它根据一系列预定义的规则替换一定的文本模式。解释器或编译器在遇到宏时会自动进行这一模式替换。对于编译语言，宏展开在编译时发生，进行宏展开的工具常被称为宏展开器。宏这一术语也常常被用于许多类似的环境中，它们是源自宏展开的概念，这包括键盘宏和宏语言。绝大多数情况下，“宏”这个词的使用暗示着将小命令或动作转化为一系列指令。 宏的用途在于自动化频繁使用的串行或者是获得一种更强大的抽象能力——但这常常是一回事。	在计算机科学中，**内联函数**（有时称作**在线函数**或**编译时期展开函数**）是一种编程语言结构，用来建议编译器对一些特殊函数进行内联扩展（有时称作**在线扩展**）；也就是说建议编译器将指定的函数体插入并取代每一处调用该函数的地方（上下文），从而节省了每次调用函数带来的额外时间开支。但在选择使用内联函数时，必须在程序占用空间和程序执行效率之间进行权衡，因为过多的比较复杂的函数进行内联扩展将带来很大的存储资源开支。另外还需要非常注意的是对递归函数的内联扩展可能带来部分编译器的无穷编译。

17、简述在面向对象程序设计中，引入继承和封装的主要作用
继承：代码重用
封装：代码安全

18、简述 C 语言中指针及其作用

19、Java 语言的多线程机制

20、简述四种常见的数据逻辑结构

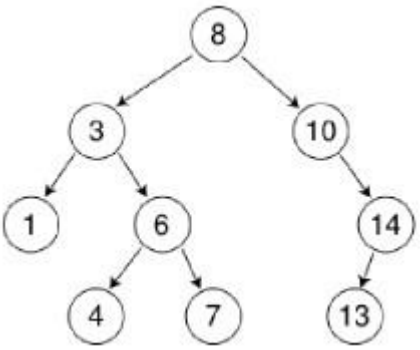
① 集合 集合中任何两个数据元素之间都没有逻辑关系，组织形式松散。

② 线性结构 线性结构中的 结点按逻辑关系依次排列形成一个“锁链”。

③ 树形结构 树形结构具有分支、层次特性，其形态有点象自然界中的树。

④ 图状结构 图状结构中的结点按逻辑关系互相缠绕，任何两个结点都可以邻接

21、简述在一棵二叉排序树中查找一特定元素 x 的算法过程



二叉排序树（Binary Sort Tree）又称二叉查找树。 它或者是一棵空树；或者是具有下列性质的二叉树：
（1）若左子树不空，则左子树上所有结点的值均小于它的根结点的值；
（2）若右子树不空，则右子树上所有结点的值均大于它的根结点的值；
（3）左、右子树也分别为二叉排序树；

22、简述快速排序的基本思想，并说明其最不理想情形

[分治法]通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。

在最坏的情况是，两子数列拥有大各为 1 和 n-1，且调用树（call tree）变成为一个 n 个嵌套（nested）调用的线性连串（chain）。第 i 次调用作了 O(n-i)的工作量，且 $\sum_{i=0}^{n}(n-i)=O(n^2)$ 递归关系式为：

$T(n) = O(n) + T(1) + T(n - 1) = O(n) + T(n - 1)$ 这与插入排序和选择排序有相同的关系式，以及它被解为 T(n) = O(n2)。它的最坏情况是很恐怖的，需要

$$\sum_{i=0}^{n}(n-i+1)=\Theta(n^2)$$

空间，远比数列本身还多。

23、简述在一带权有向图中寻找关键路径的基本思想

关键路径：关键路径是指网络终端元素的元素的序列，该序列具有最长的总工期并决定了整个项目的最短完成时间。在 AOE 网中，从始点到终点具有最大路径长度（该路径上的各个活动所持续的时间之和）的路径为关键路径
关键活动：关键路径上的活动称为关键活动。

只有所有关键活动提前完成，整个工程才能提前完成。

最早可能开始时间 Ve[i]：从原点到顶点 Vi 最长路径的长度（之前所有事情做完了才可能开始）；

最迟允许开始时间 Vl[i]：保证汇点 Vn-1 在 Ve[n-1]时刻完成的前提下（整体工期不拖），事件 Vi 最迟允许的开始时间。Vl[i]=min{Vl[k]-dur(<Vi,Vk>)}

关键活动：松弛时间(slack time)Al[j]-Ae[k]==0 的节点。

24、类作用域和文件作用域的区别是什么
文件作用域也称“全局作用域”。

- 定义在所有函数之外的标识符，具有文件作用域，作用域为从定义处到整个源文件结束。
- 文件中定义的全局变量和函数都具有文件作用域。
- 如果某个文件中说明了具有文件作用域的标识符，该文件又被另一个文件包含，则该标识符的作用域延伸到新的文件中。如 cin 和 cout 是在头文件 iostream.h 中说明的具有文件作用域的标识符，它们的作用域也延伸到嵌入 iostream.h 的文件中。

操作系统

1. 进程和线程的区别及联系，操作系统的程序栈...

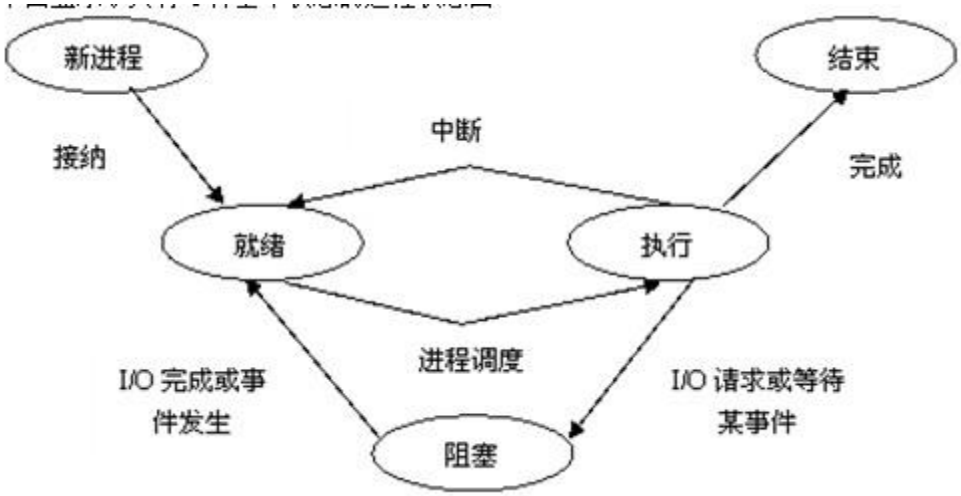
线程和进程的区别：
1、 线程是进程的一部分，所以线程有的时候被称为是轻权进程或者轻量级进程。
2、 一个没有线程的进程是可以被看作单线程的，如果一个进程内拥有多个进程，进程的执行过程不是一条线（线程）的，而是多条线（线程）共同完成的。
3、 系统在运行的时候会为每个进程分配不同的内存区域，但是不会为线程分配内存（线程所使用的资源是它所属的进程的资源），线程组只能共享资源。那就是说，出了 CPU 之外（线程在运行的时候要占用 CPU 资源），计算机内部的软硬件资源的分配与线程无关，线程只能共享它所属进程的资源。
4、 与进程的控制表 PCB 相似，线程也有自己的控制表 TCB，但是 TCB 中所保存的线程状态比 PCB 表中少多了（上下文保存一下就行）。
5、 进程是系统所有资源分配时候的一个基本单位，拥有一个完整的虚拟空间地址，并不依赖线程而独立存在。

进程与程序的区别：
程序是一组指令的集合，它是静态的实体，没有执行的含义。而进程是一个动态的实体，有自己的生命周期。一般说来，一个进程肯定与一个程序相对应，并且只有一个，但是一个程序可以有多个进程，或者一个进程都没有也可以只有一个进程。除此之外，进程还有并发性和交往性。简单地说，进程是程序的一部分，程序运行的时候会产生进程。总结：线程是进程的一部分，进程是程序的一部分。

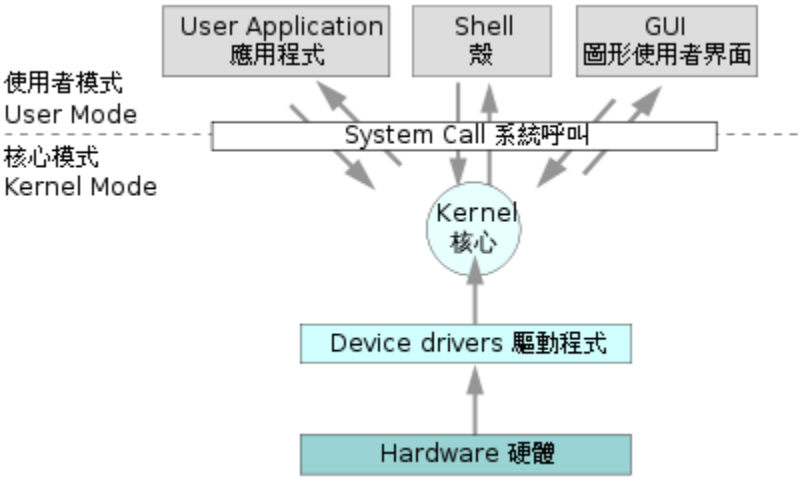
前一句说的不太准确，线程也有自己的资源，比如栈，私有数据等等。说他使用而不拥有资源指的是使用的是进程的打开文件句柄，进程的全局数据，进程的地址空间等等,这些都属于进程，而不属于线程，进程内个线程共享。

进程切换比线程切换开销大是因为进程切换时要切页表，而且往往伴随着页调度，因为进程的数据段代码段要换出去，以便把将要执行的进程的内容换进来。本来进程的内容就是线程的超集。而且线程只需要保存线程的上下文（相关寄存器状态和栈的信息）就好了，动作很小。

2. OS 里面进程的“三态”“五态”“七态”是什么



3. 什么是操作系统



操作系统（英文：**Operating System**，缩写：**OS**）是管理计算机硬件与软件资源的计算机程序，同时也是计算机系统的内核与基石。操作系统需要处理如管理与配置内存、决定系统资源供需的优先次序、控制输入与输出设备、操作网络与管理文件系统等基本事务。操作系统也提供一个让用户与系统交互的操作界面。

4. 死锁的条件，检测死锁的可能方法及其基本思想

A **deadlock** is a situation in which two or more competing actions are each waiting for the other to finish, and thus neither ever does.

A deadlock situation can arise if all of the following conditions hold simultaneously in a system:[1]

1. **Mutual Exclusion**（互斥）**:** At least two resources must be non-shareable.[1] Only one process can use the resource at any given instant of time.
2. **Hold and Wait** or **Resource Holding:** A process is currently holding at least one resource and requesting additional resources which are being held by other processes.
3. **No Preemption**（禁止抢占）**:** The operating system must not de-allocate resources once they have been allocated; they must be released by the holding process voluntarily.
4. **Circular Wait（循环等待）:** A process must be waiting for a resource which is being held by another process, which in turn is waiting for the first process to release the resource. In general, there is a set of waiting processes, $P = \{P_1, P_2, ..., P_N\}$, such that P_1 is waiting for a resource held by P_2, P_2 is waiting for a resource held by P_3 and so on until P_N is waiting for a resource held by P_1.[1][7]

These four conditions are known as the **Coffman conditions** from their first description in a 1971 article by Edward G. Coffman, Jr.[7] Unfulfillment of any of these conditions is enough to preclude a deadlock from occurring.

Ensure that the system will never enter a deadlock state.

□Prevention: Ensure one of the four conditions fails.

□Avoidance: The OS needs more information so that it can determine if the current request can be satisfied or

delayed.

死锁检测：

1. Resource-Allocation Graph
2. Detection Algorithm

5. 用户态和内核态

当程序运行在 3 级特权级上时，就可以称之为运行在用户态，因为这是最低特权级，是普通的用户进程运行的特权级，大部分用户直接面对的程序都是运行在用户态；反之，当程序运行在级特权级上时，就可以称之为运行在内核态。

用户态切换到内核态的 3 种方式

a. 系统调用

这是用户态进程主动要求切换到内核态的一种方式，用户态进程通过系统调用申请使用操作系统提供的服务程序完成工作，比如前例中 fork()实际上就是执行了一个创建新进程的系统调用。而系统调用的机制其核心还是使用了操作系统为用户特别开放的一个中断来实现，例如 Linux 的 int 80h 中断。

b. 异常

当 CPU 在执行运行在用户态下的程序时，发生了某些事先不可知的异常，这时会触发由当前运行进程切换到处理此异常的内核相关程序中，也就转到了内核态，比如缺页异常。

c. 外围设备的中断

当外围设备完成用户请求的操作后，会向 CPU 发出相应的中断信号，这时 CPU 会暂停执行下一条即将要执行的指令转而去执行与中断信号对应的处理程序，如果先前执行的指令是用户态下的程序，那么这个转换的过程自然也就发生了由用户态到内核态的切换。比如硬盘读写操作完成，系统会切换到硬盘读写的中断处理程序中执行后续操作等。

这 3 种方式是系统在运行时由用户态转到内核态的最主要方式，其中系统调用可以认为是用户进程主动发起的，异常和外围设备中断则是被动的。

6. 面包店算法

该算法的基本思想源于顾客在面包店中购买面包时的排队原理. 顾客在进入面包店前, 首先抓一个号, 然后按照号码由小到大的次序依次进入面包店购买面包. 这里, 面包店发放的号码是由小到大的, 但是两个或两个以上的顾客却有可能得到相同的号码(使所抓号码不同需要互斥), 如果多个顾客抓到相同的号码, 则规定按照顾客名字的字典次序进行排序, 这里假定顾客是没有重名的. 在计算机系统中, 顾客就相当于进程, 每个进程有一个唯一的标识, 我们用 P 的下面加一个下标来表示. 例如: 对于 Pi 和 Pj, 如果有 i<j, 则先为 Pi 服务, 即 Pi 先进入临界区

7. 系统调用和库函数的区别

(1)调用形式不同。过程（函数）使用一般调用指令，其转向地址是固定不变的，包含在跳转语句中；但系统调用中不包含处理程序入口，而仅仅提供功能号，按功能号调用。

(2)被调用代码的位置不同。过程（函数）调用是一种静态调用，调用者和被调用代码在同一程序内，经过连接编辑后作为目标代码的一部份。当过程（函数）升级或修改时，必须重新编译连结。而系统调用是一种动态调用，系统调用的处理代码在调用程序之外（在操作系统中），这样一来，系统调用处理代码升级或修改时，与调用程序无关。而且，调用程序的长度也大大缩短，减少了调用程序占用的存储空间。

(3)提供方式不同。过程（函数）往往由编译系统提供，不同编译系统提供的过程（函数）可以不同；系统调用由操作系统提供，

一旦操作系统设计好，系统调用的功能、种类与数量便固定不变了。

(4)调用的实现不同。程序使用一般机器指令（跳转指令）来调用过程（函数），是在用户态运行的；程序执行系统调用，是通过中断机构来实现，需要从用户态转变到核心态，在管理状态执行，因此，安全性好。

8. 经典进程同步问题是什么，同步思想？

生产者-消费者问题是著名的进程同步问题，它描述一组生产者进程向一组消费者进程提供消息。它们共享一个有界缓冲池，生产者向其中投放消息，消费者从中取得消息。生产者-消费者问题是许多相互合作进程的一种抽象。假定缓冲池中有 n 个缓冲区，每个缓冲区存放一个消息。由于缓冲池是临界资源，它只允许一个生产者投入消息，或者一个消费者从中取出消息。生产者之间、生产者与消费者之间、消费者之间都必须互斥地使用缓冲池。所以必须设置互斥信号量 mutex，它代表缓冲池资源，它的数值为 1。

读者-写者问题

问题描述：一个数据集（如文件）被几个并行进程所共享，有些进程只要求读数据集内容，称为读者，而另一些进程则要求修改数据集内容，称为写者，几个读者可以同时读数据集，而不需要互斥，但一个写者不能和其他进程（不管是写者或读者）同时访问这些数据集，它们之间必须互斥。

哲学家进餐问题

该问题描述如下：有五个哲学家，他们的生活方式是交替地进行思考和进餐。哲学家们公用一张圆桌，周围放有五把椅子，每人坐一把。在圆桌上有五个碗和五根筷子，当一个哲学家思考时，他不与其他人交谈，饥饿时便试图取用其左、右最靠近他的筷子，但他可能一根都拿不到。只有在他拿到两根筷子时，方能进餐，进餐完后，放下筷子又继续思考。

9. 文件管理及组织，FAT

FAT = File Allocation Table

10. 设备驱动程序是否属于 OS，他的作用是什么

不是，驱动程序是另外安装的软件，是操作系统控制并且和硬件之间通讯的桥梁（程序）

11. 程序和任务的区别

任务是最抽象的，是一个一般性的术语，指由软件完成的一个活动。一个任务既可以是一个进程,也可以是一个线程。简而言之,它指的是一系列共同达到某一目的的操 作。例如,读取数据并将数据放入内存中。这个任务可以作为一个进程来实现,也可以作为一个线程（或作为一个中断任务）来实现。 **进程**常常被定义为程序的执行。可以把一个进程看成是一个独立的程序,在内存中有其完备的数据空间和代码空间。一个进程所拥有的数据和变量只属于

它自己。 **线程**则是某一进程中一路单独运行的程序。也就是说，线程存在于进程之中。一个进程由一个或多个线程构成，各线程共享相同的代码和全局数据，但各有其自己的堆栈。由于堆栈是每个线程一个，所以局部变量对每一线程来说是私有的。由于所有线程共享同样的代码和全局数据，它们比进程更紧密，比单独的进程间更趋向于相互作用,线程间的相互作用更容易些,因为它们本身就有某些供通信用的共享内存：进程的全局数据进程的全局数据进程的全局数据进程的全局数据

进程：资源分配、调度运行的基本单位。
线程：进程中执行运算的最小单位，执行处理机调度的基本单位。

12. 数据库安全性和操作系统安全性的关系

安全性问题不是数据库系统所独有的,所有计算机系统都有这个问题.只是在数据库系统中大量数据集中存放,而且为许多最终用户直接共享,从而使安全性问题更为突出.系统安全保护措施是否有效是数据库系统的主要指标之一.数据库的安全性和计算机系统的安全性,包括操作系统,网络系统的安全性是紧密联系,相互支持的.

13. 中断处理的过程

请求中断→响应中断→关闭中断→保留断点→中断源识别→保护现场→中断服务子程序→恢复现场→中断返回

在实际运行中，一旦设备通过某引脚 N 向 8259A **发出中断指令**，后者便向 8086A 的 INTR 引脚**发送中断信号**。8086A 通过 INTA 引脚通知 8259A 中断有效（这个过程实际上还包括对此 8259A 的选址），后者即通过地址总线将对应引脚 N 的**中断类型码**（已预先存好，见上节）发送给 CPU。CPU 得到中断类型码后，先进行**现场保护**，主要包括：

1. 状态寄存器 FLAGS 压栈（同时堆栈寄存器 SP-2）；
2. 关闭中断（将 FLAGS 寄存器的 IF 位置零）；
3. 将当前代码段寄存器 CS 和程序计数器 IP 压栈（同时堆栈寄存器 SP-4）。

现场保护完成后，CPU 开始按照前述的两步骤翻译中断程序入口地址。在得到中断处理程序地址之后但调用中断处理程序之前，CPU 会再**检查一下 NMI 引脚是否有信号**，以防在刚才的处理过程中忽略了可能的 NMI 中断。NMI 的优先级始终高于 INTR。

中断处理程序虽然是由程序员编写，但须循一定规范。作为例程，中断处理程序应该先将各寄存器信息（除了 IP 和 CS，此二寄存器现已指向当前中断程序）压入堆栈予以保存，这样才能在中断处理程序内部使用这些寄存器。在程序结束时，应该按与压栈保护时相反的顺序弹出各寄存器的值。中断程序的最后一句始终是 IRET 指令，这条指令将栈顶 6 个字节分别弹出并存入 IP、CS 和 FLAGS 寄存器，完成了现场的还原。

当然，如果是操作系统的中断处理程序，则未必——通常不会——还原中断前的状态。这样的中断处理程序通常会在调用完寄存器保存例程后，调用进程调度程序（多由高级语言编写），并决定下一个运行的进程。随后将此进程的寄存器信息（上次中断时保存下来的）存入寄存器并返回。在中断程序结束之后，主程序也发生了改变。

14. 计算机系统怎样实现存储保护

1. 防止地址越界（对进程所产生的地址必须加以检查，发生越界时产生中断，由操作系统进行相应处理）
2. 防止操作越权（对属于自己区域的信息，可读可写：对公共区域中允许共享的信息或获得授权可使用的信息，可读而不可修改；对未授权使用的信息，不可读，不可写）

15. 调度的基本准则(Scheduling Criteria)

There are many criteria for comparing different scheduling algorithms. Here are five common ones:

- CPU utilization(CPU 利用率)
- Throughput（吞吐量）
- Turnaround time（周转时间）
- Waiting time（等待时间）
- Response time（响应时间）

16. 多线程是否真正能提高效率

17. 磁盘调度算法有哪几种

FCFS

SSTF (Shortest Seek Time First)

Scan/Look

C-Sacn/C-Look

18. RAID 工作原理

RAID(Redundant Array of Independent Disks)通过条带化存储和奇偶校验两个措施来实现其冗余和容错的目标。条带化存储意味着可以一次写入一个数据块的方式将文件写入多个磁盘。条带化存储技术将数据分开写入多个驱动器，从而提高数据传输速率并缩短磁盘处理总时间。这种系统非常适用于交易处理、但可靠性却很差，因为系统的可靠性等于最差的单个驱动器的可靠性。

奇偶校验通过在传输后对所有数据进行冗余校验可以确保数据的有效性。利用奇偶校验，当 RAID 系统的一个磁盘发生故障时，其它磁盘能够重建该故障磁盘。在这两种情况中，这些功能对于操作系统都是透明的。由磁盘阵列控制器（DAC）进行条带化存储和奇偶校验控制。

19. Windows 中段最长多少字节

?

20. 同步、互斥

21. 简述常用的进程通信方式及其基本思想（至少两种）

Method	Short Description	Provided by (operating systems or other environments)
File	A record stored on disk that can be accessed by name by any process	Most operating systems
Signal	A system message sent from one process to another, not usually used to store information but instead give commands.	Most operating systems; some systems, such as Windows, implement signals in only the C run-time library and provide no support for their use as an IPC method[citation needed]

Socket	A data stream sent over a network interface, either to a different process on the same computer or to another computer	Most operating systems
Message queue	An anonymous data stream(匿名数据流) similar to the pipe, but stores and retrieves information in packets.	Most operating systems
Pipe	A two-way data stream interfaced through standard input and outputand is read character by character.	All POSIX systems, Windows
Named pipe	A pipe implemented through a file on the file system instead ofstandard input and output.	All POSIX systems, Windows
Semaphore	A simple structure that synchronizes threads or processes acting on shared resources.	All POSIX systems, Windows
Shared memory	Multiple processes given access to the same memory, allowing all to change it and read changes made by other processes.	All POSIX systems, Windows
Message passing(shared nothing)	Similar to the message queue.	Used in MPI paradigm, Java RMI, CORBA, DDS, MSMQ, MailSlots, QNX, others
Memory-mapped file	A file mapped to RAM and can be modified by changing memory addresses directly instead of outputting to a stream, shares same benefits as a standard file.	All POSIX systems, Windows

22. 简述请求段页式虚拟内存管理基本思想

Bring a page into memory **only when it is needed.**

Less I/O needed
Less memory needed
Faster response
More users
Page is needed ⇒ reference to it
invalid reference ⇒ abort
not-in-memory ⇒ bring to memory

网络

1. 路由协议

常见路由协议有：RIP，OSPF，BGP 等

路由信息协议（英语：Routing Information Protocol，缩写：**RIP**）是一种使用最广泛的内部网关协议（IGP）。（IGP）是在内部网络上使用的路由协议(在少数情形下,也可以用于连接到因特网的网络)，它可以通过不断的交换信息让路由器动态的适应网络连接的变化，这些信息包括每个路由器可以到达哪些网络，这些网络有多远等。 RIP 属于网络层。RIP 所使用的路由算法是 Bellman-Ford 算法.这种算法最早被用于一个计算机网络是在 1969 年，当时是作为 ARPANET 的初始路由算法。

开放式最短路径优先（英文：Open Shortest Path First，OSPF）是对链路状态路由协议的一种实现，隶属内部网关协议（IGP），故运作于自治系统内部。著名的迪克斯加算法被用来计算最短路径树。它使用“代价（Cost）”作为路由度量。链路状态数据库（LSDB）用来保存当前网络拓扑结构，它在同一区域中的所有路由器上是相同的。

边界网关协议（ 英文：Border Gateway Protocol, BGP）是互联网上一个核心的去中心化自治路由协议。它通过维护 IP 路由表或‘前缀’表来实现自治系统（AS）之间的可达性，属于矢量路由协议。 BGP 不使用传统的内部网关协议（IGP）的指标，而使用基于路径、网络策略或规则集来决定路由。因此，它更适合被称为矢量性协议，而不是路由协议。

2. CSMA/CD, 指数回退

载波侦听多路访问／冲突检测（英语：Carrier Sense Multiple Access with Collision Detection，CSMA/CD）

此方案要求设备在发送帧的同时要对信道进行侦听，以确定是否发生冲突，若在发送数据过程中检测到冲突，则进行如下冲突处理操作：

1. 发送特殊阻塞信息并立即停止发送数据：特殊阻塞信息是连续几个字节的全 1 信号，此举意在强化冲突，以使得其它设备能尽快检测到冲突发生。
2. 在固定时间（一开始是 1 contention period times）内等待随机的时间，再次发送。
3. 若依旧碰撞，则采用截断二进制指数避退算法进行发送。即十次之内停止前一次“固定时间”的两倍时间内随机再发送，十次后则停止前一次“固定时间”内随机再发送。尝试 16 次之后仍然失败则放弃传送。

载波侦听多路访问／冲突避免（英语：Carrier Sense Multiple Access with Collision Avoidance，**CSMA/CA**）

此种方案采用主动避免碰撞而非被动侦测的方式来解决冲突问题。可以满足那些不易准确侦测是否有冲突发生的需求，如无线网域。

CSMA/CA 协议主要使用两种方法来避免碰撞：

1. 设备欲发送讯框(Frame)，且讯框听到通道空闲时，维持一段时间后，再等待一段随机的时间依然空闲时，才提交数据。由于各个设备的等待时间是分别随机产生的，因此很大可能有所区别，由此可以减少冲突的可能性。
2. RTS-CTS 三向握手（英语：handshake）：设备欲发送讯框前，先发送一个很小的 RTS（Request to Send）讯框给目标端，等待目标端回应 CTS（Clear to Send）帧后，才开始传送。此方式可以确保接下来传送数据时，不会发生冲突。同时由于 RTS 帧与 CTS 帧都很小，让传送的无效开销变小。

3. 解释 FTP、HTTP 的全称及其原理。FTP、HTTP 文件传输的异同

   FTP：File Transfer Protocol

   HTTP: Hyper Text Transfer Protocol

   都是应用层协议。

4. 七层协议的名称

   Physical

   Data Link

   Network

   Transmission

   Session

   Presentation

   Application

5. 电子邮件发送到接收的过程，协议

   电子邮件的工作过程遵循客户-服务器模式。每份电子邮件 的发送都要涉及到发送方与接收方，发送方式构成客户端，而接收方构成服务器，服务器含有 众多用户的电子信箱。发送方通过邮件客户程序，将编辑好的电子邮件向邮局服务器（SMTP 服 务器）发送。邮局服务器识别接收者的地址，并向管理该地址的邮件服务器（POP3 服务器）发 送消息。邮件服务器识将消息存放在接收者的电子信箱内，并告知接收者有新邮件到来。接收 者通过邮件客户程序连接到服务器后，就会看到服务器的通知，进而打开自己的电子信箱来查 收邮件。

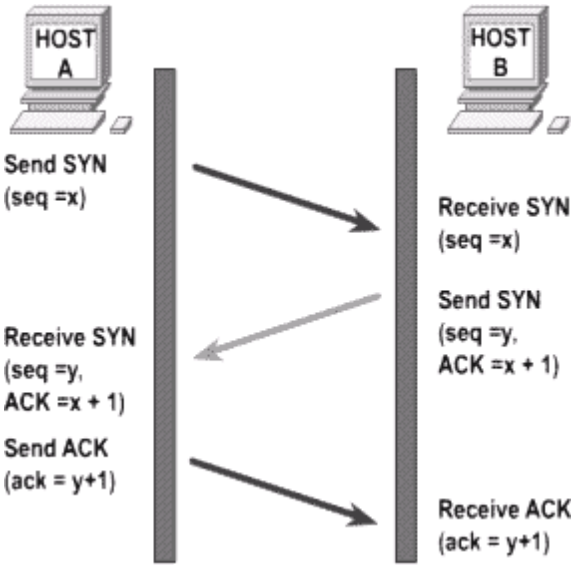
   发送：SMTP

   接收：POP, IMAP

6. P2P 技术，具体的实现机制

   实现 P2P 需要一个中转服务器。也就是需要一个第三方。（一会儿我们来说为什么需要一个第三方）

7. 网络中的握手问题



第一次握手：建立连接时，客户端发送 syn 包(syn=j)到服务器，并进入 SYN_SEND 状态，等待服务器确认；

第二次握手：服务器收到 syn 包，必须确认客户的 SYN（ack=j+1），同时自己也发送一个 SYN 包（syn=k），即 SYN+ACK 包，此时服务器进入 SYN_RECV 状态；

第三次握手：客户端收到服务器的 SYN+ACK 包，向服务器发送确认包 ACK(ack=k+1)，此包发送完毕，客户端和服务器进入 ESTABLISHED 状态，完成三次握手。

完成三次握手，客户端与服务器开始传送数据

8. 无线传感网络(WSN)及其应用

无线传感网络（Wireless sensor network），或译无线感知网络，是由许多在空间中分布的自动装置组成的一种无线通讯计算机网络，这些装置使用传感器协作地监控不同位置的物理或环境状况（比如温度、声音、振动、压力、运动或污染物）。无线传感器网络的发展最初起源于战场监测等军事应用。而现今无线传感器网络被应用于很多民用领域，如环境与生态监测、健康监护、家居自动化以及交通控制等。

无线传感网络主要包括三个方面：感应，通讯，计算（硬件，软件，算法）。其中的关键技术主要有无线数据库技术，比如使用在无线传感器网络的查询，和用于和其他传感器通讯的网络技术，特别是多次跳跃路由协议。

9. 当前网络新技术及其应用,Web2.0

Web 2.0，指的是一个利用 Web 的平台，由**用户主导**而生成的内容互联网产品模式，为了区别传统由网站雇员主导生成的内容而定义为 web2.0

10. ipV4 与 ipV6 的区别

目前的全球因特网所采用的协议族是 TCP/IP 协议族。IP 是 TCP/IP 协议族中网络层的协议，是 TCP/IP 协议族的核心协议。目前 IP 协议的版本号是 4(简称为 IPv4)，发展至今已经使用了 30 多年。

IPv4 的地址位数为 32 位，也就是最多有 2 的 32 次方的电脑可以联到 Internet 上。

近十年来由于互联网的蓬勃发展，IP 位址的需求量愈来愈大，使得 IP 位址的发放愈趋严格，各项资料显示全球 IPv4 位址可能在 2005 至 2008 年间全部发完。

什么是 IPv6?

IPv6 是下一版本的互联网协议，也可以说是下一代互联网的协议，它的提出最初是因为随着互联网的迅速发展，IPv4 定义的有限地址空间将被耗尽，地址空间的不足必将妨碍互联网的进一步发展。为了扩大地址空间，拟通过 IPv6 重新定义地址空间。IPv6 采用 128 位地址长度，几乎可以不受限制地提供地址。按保守方法估算 IPv6 实际可分配的地址，整个地球的每平方米面积上仍可分配 1000 多个地址。在 IPv6 的设计过程中除了一劳永逸地解决了地址短缺问题以外，还考虑了在 IPv4 中解决不好的其它问题，主要有端到端 IP 连接、服务质量（QoS）、安全性、多播、移动性、即插即用等。

IPv6 与 IPv4 相比有什么特点和优点？

更大的地址空间。IPv4 中规定 IP 地址长度为 32，即有 2^32-1 个地址；而 IPv6 中 IP 地址的长度为 128，即有 2^128-1 个地址。

更小的路由表。IPv6 的地址分配一开始就遵循聚类(Aggregation)的原则，这使得路由器能在路由表中用一条记录(Entry)表示一片子网，大大减小了路由器中路由表的长度，提高了路由器转发数据包的速度。

增强的组播(Multicast)支持以及对流的支持(Flow-control)。这使得网络上的多媒体应用有了长足发展的机会，为服务质量(QoS)控制提供了良好的网络平台.

加入了对自动配置(Auto-configuration)的支持。这是对 DHCP 协议的改进和扩展，使得网络(尤其是局域网)的管理更加方便和快捷.

更高的安全性.在使用 IPv6 网络中用户可以对网络层的数据进行加密并对 IP 报文进行校验,这极大的增强了网络安全.

11. TCP 拥塞控制(congestion control)与流量控制(flow control)的功能和区别

流量控制解决的问题：通信双方速率不匹配。方法：Sliding window，控制滑动窗口大小，自适应速率。

拥塞控制解决的问题：双方速率都允许的情况下，如何保证途中不出问题。方法：使用 RTT(Round Trip Time)以及 RTO(Retransmission Time Out)以及计算这两个量的算法，保证传输可能失败的时候重传。

12. PPP 协议

点对点协议（英语：Point-to-Point Protocol，PPP）工作在数据链路层（以 OSI 参考模型的观点）。它通常用在两节点间创建直接的连接，并可以提供连接认证、传输加密（使用 ECP，RFC 1968）以及压缩。

13. 集线器、路由器和交换机有什么区别。中继器、集线器、交换机、网桥、网关、路由器的功能

集线器：是指将多条以太网双绞线或光纤集合连接在同一段物理介质下的设备。集线器是运作在 OSI 模型中的**实体层**。它可以视作多端口的中继器，若它侦测到碰撞，它会提交阻塞信号。

路由器（**Router**，又称**路径器**）是一种计算机网络设备，它能将数据包通过一个个网络传送至目的地（选择数据的传输路径），这个过程称为路由。路由工作在 OSI 模型的第三层——即网络层

交换机工作于 OSI 参考模型的第二层，即数据链路层。交换机内部的 CPU 会在每个端口成功连接时，通过 ARP 协议学习它的 MAC 地址，保存成一张 ARP 表。在今后的通讯中，发往该 MAC 地址的数据包将仅送往其对应的端口，而不是所有的端口。因此，交换机可用于划分数据链路层广播，即冲突域；但它不能划分网络层广播，即广播域。

桥接器（英语：**Bridge**），又称**网桥**，根据 MAC 分区块，可隔离碰撞。桥接器将网络的多个网段在数据链路层（OSI 模型第 2 层）连接起来（即桥接）。桥接器在功能上与集线器等其他用于连接网段的设备类似，不过后者工作在物理层（OSI 模型第 1 层）。

网关（英语：Gateway；台湾、港澳作**闸道器**），区别于路由器（由于历史的原因，许多有关 TCP/IP 的文献曾经把网络层使用的路由器（英语：Router）称为网关。 网关也经常指把一种协议转成另一种协议的设备，比如语音网关。**网关中并没有路由表**，他只能按照预先设定的不同网段来进行转发。网关最重要的一点就是端口映射，子网内用户在外网看来只是外网的 IP 地址对应着不同的端口，这样看来就会保护子网内的用户。

14. P2P 网络编程的特点

？

15. DNS 的递归查询和迭代查询

（1）递归查询

递归查询是一种 DNS 服务器的查询模式，在该模式下 DNS 服务器接收到客户机请求，必须使用一个准确的查询结果回复客户机。如果 DNS 服务器本地没有存储查询 DNS 信息，那么该服务器会询问其他服务器，并将返回的查询结果提交给客户机。

（2）迭代查询

DNS 服务器另外一种查询方式为迭代查询，DNS 服务器会向客户机提供其他能够解析查询请求的 DNS 服务器地址，当客户机发送查询请求时，DNS 服务器并不直接回复查询结果，而是告诉客户机另一台 DNS 服务器地址，客户机再向这台 DNS 服务器提交请求，依次循环直到返回查询的结果为止。

16. ARP 协议、ARP 攻击

地址解析协议（**A**ddress **R**esolution **P**rotocol），其基本功能为通过目标设备的 IP 地址，查询目标设备的 MAC 地址，以保证通信的顺利进行。它是 IPv4 中网络层必不可少的协议，不过在 IPv6 中已不再适用，并被邻居发现协议（NDP）所替代。

在以太网协议中规定，同一局域网中的一台主机要和另一台主机进行直接通信，必须要知道目标主机的 MAC 地址。而在 TCP/IP 协议中，网络层和传输层只关心目标主机的 IP 地址。这就导致在以太网中使用 IP 协议时，数据链路层的以太网协议接到上层 IP 协议提供的数据中，只包含目的主机的 IP 地址。于是需要一种方法，根据目的主机的 IP 地址，获得其 MAC 地址。这就是 ARP 协议要做的事情。所谓**地址解析（address resolution）**就是主机在发送帧前将目标 IP 地址转换成目标 MAC 地址的过程。

ARP 欺骗的运作原理是由攻击者发送假的 ARP 数据包到网络上，尤其是送到网关上。其目的是要让送至特定的 IP 地址的流量被错误送到攻击者所取代的地方。因此攻击者可将这些流量另行转送到真正的闸道（被动式数据包嗅探，passive sniffing）或是篡改后再转送（中间人攻击，man-in-the-middle attack）。

17. 计算机网络的接入类型有哪些

局域网、城域网、广域网和互联网四种

18. 分组交换

分组交换（**Packet switching**）在计算机网络和通讯中是一种相对于电路交换的通信范例，分组（又称消息、或消息碎片）在节点间单独路由，不需要在传输前先建立通信路径。

19. 电路交换和报文交换的区别

报文交换（英语：Message switching），又称存储转发交换，是数据交换的三种方式之一，报文整个地发送，一次一跳。报文交换是分组交换的前身，是由由列奥纳德·克莱因饶克于 1961 年提出的。

报文交换的主要特点是：存储接受到的报文，判断其目标地址以选择路由，最后，在下一跳路由空闲时，将数据转发给下一跳路由。报文交换系统现今都由分组交换或电路交换网络所承载。

电路交换（Circuit Switching）是相对于封包交换(或称分组交换)的一个概念。电路交换要求必须首先在通信双方之间建立连接通道。在连接建立成功之后，双方的通信活动才能开始。通信双方需要传递的信息都是通过已经建立好的连接来进行传递的，而且这个连接也将一直被维持到双方的通信结束。在某次通信活动的整个过程中，这个连接将始终占用着连接建立开始时，通信系统分配给它的资源（通道、带宽、时隙、码字等等），这也体现了电路交换区别于分组交换的本质特征。

20.

计组/接口

1. Cache 的两种更新策略

一般有两种回写策略：写回（Write back）和写通（Write through）。

写回是指，仅当一个缓存块需要被替换回内存时，才将其内容写入内存。如果缓存命中，则总是不用更新内存。为了减少内存写操作，缓存块通常还设有一个脏位（dirty bit），用以标识该块在被载入之后是否发生过更新。如果一个缓存块在被置换回内存之前从未被写入过，则可以免去回写操作。

2. 计算机中小数点是如何表示的
定点、浮点表示。

3. 计算机中如何表示数据，知识
？

4. Cache 的原理及思想，评价标准，改进方案，计算机软硬件中其他用到这个思想的方法

缓存之所以有效，主要是因为程序运行时对内存的访问呈现局部性（Locality）特征。这种局部性既包括空间局部性（Spatial Locality），也包括时间局部性（Temporal Locality）。有效利用这种局部性，缓存可以达到极高的命中率。

评估缓存的性能通常使用平均内存访问时间（Average Memory Access Time, AMAT）这一指标。

5. 分页、分段、段页式的特点，为什么要引入

分页（英语：Paging），是一种操作系统里存储器管理的一种技术，可以使电脑的主存可以使用存储在辅助存储器中的数据。操作系统会将辅助存储器中的数据分区成固定大小的区块，称为“页”（pages）。当不需要时，将分页由主存移到辅助存储器；当需要时，再将数据取回，加载主存中。相对于分段，分页允许存储器存储于不连续的区块以维持文件系统的整齐。在分页实现前，分段会使文件于系统中连续，这使它容易造成空间杂乱且产生许多碎片。

储存单位划分原则不同：分页存储是信息的物理单位，为提高内存利用率设置；分段存储是信息的逻辑单位，为满足用户需要设置。

存储单位特征不同：分页存储的存储单位由系统确定，大小相等；分段存储的存储单位由用户程序确定，大小不等。

逻辑地址性质不同：分页存储是页号，页内地址，单一的线性地址；分段存储是段号，段内地址，二维地址。

段页式存储器兼顾了程序和数据的逻辑结构以及存储器的物理结构两方面的特点，兼收页式和段式虚拟存储器两者的长处，并具有如下特点：

面向用户的程序地址空间采用段式分隔。
内存分成位置固定、长度相等的若干页面。
将每一段的线性地址空间划分为页，页长与内存页面相等。

6. 中断的作用

中断的典型应用包括系统时钟、磁盘输入输出操作、断电信号以及软件自陷等。

7. DMA 优先级为什么比 CPU 高

DMA 是有专用数据通路的
DMA 传送过程中 cpu 可以继续工作
不会阻塞 可以响应外部中断
独享总线的 DMA 会在传输是占用总线,这时候 CPU 将总线让给 DMA,这时候 DMA 优先级比 CPU 高
请求型 DMA 会在占用总线是向 CPU 发起请求,CPU 优先级比 DMA 高

8. 虚拟内存容量由什么决定

寻址长度（地址总线数量）
虚拟存储区的容量与物理主存大小无关，而受限于计算机的地址结构和可用磁盘容量。

9. 根据内存大小算地址总线数量

同上

10. 8086 指令地址的计算方式

基址寄存器 BX, BP
变址寄存器 SI, DI
BX,SI,DI 默认 DS 段
BP 默认 SS 段
立即数寻址
直接寻址： MOV AX, [2000H] MOV AX, ES:[2000H]
寄存器寻址 MOV AX, BX
寄存器间接寻址 MOV AX, [BX]
寄存器相对寻址 MOV AX, [SI+06H]
基址变址寻址 MOV AX, [BS+SI]
基址变址相对寻址 MOV AX, [BX+DI+6]

11. 冯.诺伊曼机以什么部件为中心

以运算器为中心，I/O 设备与存储器间的数据传送都要经过运算器

12. CISC/RISC 的特点

（1） 指令系统：RISC 设计者把主要精力放在那些经常使用的指令上，尽量使它们具有简单高效的特色。对不常用的功能，常通过组合指令来完成。因此，在 RISC 机器上实现特殊功能时，效率可能较低。但可以利用流水技术和超标量技术加以改进和弥补。而 CISC 计算机的指令系统比较丰富，有专用指令来完成特定的功能。因此，处理特殊任务效率较高。

（2） 存储器操作：RISC 对存储器操作有限制，使控制简单化；而 CISC 机器的存储器操作指令多，操作直接。

（3） 程序：RISC 汇编语言程序一般需要较大的内存空间，实现特殊功能时程序复杂，不易设计；而 CISC 汇编语言程序编程相对简单，科学计算及复杂操作的程序社设计相对容易，效率较高。

（4） 中断：RISC 机器在一条指令执行的适当地方可以响应中断；而 CISC 机器是在一条指令执行结束后响应中断。

（5） CPU：RISC CPU 包含有较少的单元电路，因而面积小、功耗低；而 CISC CPU 包含有丰富的电路单元，因而功能强、面积大、功耗大。

（6） 设计周期：RISC 微处理器结构简单，布局紧凑，设计周期短，且易于采用最新技术；CISC 微处理器结构复杂，设计周期长。

（7） 用户使用：RISC 微处理器结构简单，指令规整，性能容易把握，易学易用；CISC 微处理器结构复杂，功能强大，实现特殊功能容易。

（8） 应用范围：由于 RISC 指令系统的确定与特定的应用领域有关，故 RISC 机器更适合于专用机；而 CISC 机器则更适合于通用机。

13. 8254,8255,8259

14. .汇编的，说是怎么让寄存器低两位的值取反

15. 一级缓存存取速度快是为什么

静态 RAM(SRAM)不用刷新，直接与 CPU 数据总线相连

16. RS-232 接口

17. 大意是说 汇编语言 结果为零时 ZF 是多少（各个 Flag 的含义）

1

18. 主存-Cache 结构有什么作用

19. 简述虚拟存储器的基本概念

虚拟存储器是指具有请求调入功能和置换功能，能从逻辑上对内存容量加以扩充的一种存储器系统。其逻辑容量由内存容量和外存容量之和来决定，其运行速度接近于内存速度，而每位的成本却又接近于外存。可见，虚拟存储技术是一种性能非常优越的存储器管理技术，故被广泛地应用于大、中、小型和微型机器中。

20. 简述微程序控制器的基本思想

微程序控制的基本思想，就是仿照通常的解题程序的方法，把操作控制信号编成所谓的“微指令”，存放到一个只读存储器里．当机器运行时，一条又一条地读出这些微指令，从而产生全机所需要的各种操作控制信号，使相应部件执行所规定的操作 ．

微程序控制的优点

微程序控制技术可代替直接由硬件连线的控制技术。由于微程序控制方法规整性好，灵活方便，通用性强，因此在大型复杂的数字系统设计中广泛应用，成为控制器的主流设计方法。

21.

其他

（广）什么是计算机，计算，语法，语义，语用

计算机（computer）俗称电脑，是一种用于高速计算的电子计算机器，可以进行数值计算，又可以进行逻辑计算，还具有存储记忆功能。

语义：机器对自然语言的理解

语用：如何运用和理解语言

（离散）群中 Lagrange 定理及其证明

根本不懂

（离散）解释元素，树，图。并各举一例

（软工）软件架构的理解

软件架构是有关软件整体结构与组件的抽象描述，用于指导大型软件系统各个方面的设计。

软件架构师定义和设计软件的模块化，模块之间的交互，用户界面风格，对外接口方法，创新的设计特性，以及高层事物的对象操作、逻辑和流程。

软件架构师与客户商谈概念上的事情，与经理商谈广泛的设计问题，与软件工程师商谈创新的结构特性，与程序员商谈实现技巧，外观和风格。

软件架构是一个系统的草图。软件架构描述的对象是直接构成系统的抽象组件。各个组件之间的连接则明确和相对细致地描述组件之间的通讯。在实现阶段，这些抽象组件被细化为实际的组件，比如具体某个类或者对象。在面向

对象领域中，组件之间的连接通常用接口来实现。

（广）什么是平台无关性

就是说（java）写的程序不用修改就可以在不同的软硬件平台上运行
比如你写了一个程序，在 windows 下可以执行
那你直接拷到 Linux 下也可以执行，只要都装了 JRE

（数据库）查询优化有哪些

基于关系代数等价变换规则的优化方法：代数优化
物理优化：基于规则的启发式优化，基于代价估算的优化，两者结合的优化方法

（离散）握手定理

握手定理：有 n 个人握手，每人握手 x 次，握手总次数为 S，必有 S= nx/2。
顶点度数之和为边数之和的两倍。
推论 任何图(无向的或有向的)中，奇度顶点的个数是偶数

（数据库）事务的四个特点（ACID）

ACID，是指数据库管理系统（DBMS）在写入/异动资料的过程中，为保证事务(transaction)是正确可靠的，所必须具备的四个特性：原子性（Atomicity，或称不可分割性）、一致性（Consistency）、隔离性（Isolation，又称独立性）、持久性（Durability）。

- 原子性：一个事务(transaction)中的所有操作，**要么全部完成，要么全部不完成**，不会结束在中间某个环节。事务在执行过程中发生错误，会被回滚（Rollback）到事务开始前的状态，就像这个事务从来没有执行过一样。
- 一致性：**在事务开始之前和事务结束以后，数据库的完整性没有被破坏**。这表示写入的资料必须完全符合所有的默认规则，这包含资料的精确度、串联性以及后续数据库可以自发性地完成预定的工作。
- 隔离性：**当两个或者多个事务并发访问（此处访问指查询和修改的操作）数据库的同一数据时所表现出的相互关系**。事务隔离分为不同级别，包括读未提交(Read uncommitted)、读提交（read committed）、可重复读（repeatable read）和串行化（Serializable）。
- 持久性：在事务**完成以后，该事务对数据库所作的更改便持久地保存在数据库之中，并且是完全的**。

（数据库）基本的关系操作有哪些

关系模型中常用的关系操作包括查询操作和插入、删除、修改操作两大部分。
关系的查询表达能力很强，是关系操作中最主要的部分。查询操作可以分为： 选择、投影、连接、除、并、差、交、笛卡尔积等。
其中，选择、投影、并、差、笛卡尔积是五种基本操作。

（数据库）数据库故障的种类

1、 事务内部的故障2、系统故障3.介质故障4.计算机病毒

（数据库）SQL 的主键约束和唯一约束有什么区别

主键不能为空
而唯一可以为空
相同的就是 都不允许重复

（数据库）2PL 协议
1) 两段锁协议是指所有事务必须分两个阶段对数据项加锁和解锁。
1)在对任何数据进行读、写操作之前，首先要申请并获得对该数据的封锁；
2)在释放一个封锁之后，事务不再申请和获得任何其他封锁。
“两段”的含义是，事务分为两个阶段：第一阶段是获得封锁，也称为扩展阶段。在这阶段，事务可以申请获得任何数据项上的任何类型的锁，但是不能释放任何锁。第二阶段是释放封锁，也称为收缩阶段。在这阶段，事务释放已经获得的锁，但是不能再申请任何锁。

（数据库）关系完整性包括哪三个方面
1. 实体完整性 2.参照完整性 3.用户定义的完整性

（数据库）数据库恢复策略有哪几种
1. 数据转储（数据冗余）2.日志文件、

（？）数据字典通常包含 5 个部分
1.数据项 2.数据结构 3.数据流 4.数据存储 5.处理过程

（数据库）视图（View）的作用及优点

（数据库）数据库三要素
数据模型的三要素： 一般而言，数据模型是一组严格定义的概念的集合。这些概念精确地描述了系统的静态特征（数据结构）、动态特征（数据操作）和完整性约束条件，这就是数据模型的三要素。

（数据库）索引技术、日志文件
目的：提供多种存储路径，加快查找速度。
建立索引需要考虑的问题：1.没有查询、统计的需要则不建 2.数据增删改频繁，系统会花费许多时间来维护索引，从而降低了查询效率

（数据库）完整性与安全性的区别
完整性和安全性是两个不同的概念。前者是为了防止数据库中存在不符合语义的数据，防止错误信息的输入和输出造成的无效操作和错误结果，而后者是防止数据库被恶意的破坏和非法的存取。当然，完整性和安全性是密切相关的。特别是从系统实现的方法来看，某一种机制常常既可以用于安全保护亦可用于完整性保证。

（离散）无向图中某条边出现在所有生成树中的充要条件
这条边是割边（删去这条边后，连通分支会增加）

（编译）LR(0)

（离散）一棵树有 1 个度为 4 的结点，1 个度为 3 的结点，1 个度为 2 的结点，其余全为叶子结点，问一共有多少个叶子？
5 个

（编译/离散）闭包运算
数学中，对一个集合的成员进行某种运算，生成的仍然是这个集合的成员，则该集合被称为在某个运算下闭合。 例如，实数在减法下闭合，但自然数不行：自然数 3 和 7 的减法 3 − 7 的结果不是自然数。

（离散）17 条边的简单图最少多少个顶点

n 阶完全图的边数 n(n-1)/2，完全图的度是最大的

n>=7

（软工）白盒测试、黑盒测试

黑盒测试也称功能测试或数据驱动测试，它是在已知产品所应具有的功能，通过测试来检测每个功能是否都能正常使用，在测试时，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，测试者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锯而产生正确的输出信息，并且保持外部信息（如数据库或文件）的完整性。

白盒测试也称结构测试或逻辑驱动测试，它是知道产品内部工作过程，可通过测试来检测产品内部动作是否按照规格说明书的规定正常进行，按照程序内部的结构测试程序，检验程序中的每条通路是否都有能按预定要求正确工作，而不顾它的功能，白盒测试的主要方法有逻辑驱动、基路测试等，主要用于软件验证。

灰盒测试，确实是介于二者之间的，可以这样理解，灰盒测试关注输出对于输入的正确性，同时也关注内部表现，但这种关注不象白盒那样详细、完整，只是通过一些表征性的现象、事件、标志来判断内部的运行状态，有时候输出是正确的，但内部其实已经错误了，这种情况非常多，如果每次都通过白盒测试来操作，效率会很低，因此需要采取这样的一种灰盒的方法。

（软工）什么是软件测试，软件测试的目的

软件测试的经典定义是：在规定的条件下对程序进行操作，以发现程序错误，衡量软件质量，并对其是否能满足设计要求进行评估的过程。

软件测试（英语：**software testing**），描述一种用来促进鉴定软件的正确性、完整性、安全性和质量的过程。

（离散）独异点的*运算表包涵什么

幺半群是一个带有二元运算 $*: M \times M \to M$ 的集合 M，其符合下列公理：

- 结合律：对任何在 M 内的 a、b、c， $(a*b)*c = a*(b*c)$ 。
- 单位元：存在一在 M 内的元素 e，使得任一于 M 内的 a 都会符合 $a*e = e*a = a$ 。

通常也会多加上另一个公理：

- 封闭性：对任何在 M 内的 a 、 b， $a*b$ 也会在 M 内。

但这不是必要的，因为在二元运算中即内含了此一公理。

另外，幺半群也可以说是带有单位元的半群。

幺半群除了没有逆元素之外，满足其他所有群的公理。因此，一个带有逆元素的幺半群和群是一样的。

（编译）词法分析、语法分析的作用

词法分析（英语：**lexical analysis**）是计算机科学中将字符序列转换为单词（Token）序列的过程。进行词法分析的程序或者函数叫作**词法分析器**（Lexical analyzer，简称 Lexer），也叫**扫描器**（Scanner）。

语法分析器（Parser）通常是作为编译器或解释器的组件出现的，它的作用是进行语法检查、并构建由输入的单词组成的数据结构（一般是语法分析树、抽象语法树等层次化的数据结构）。语法分析器通常使用一个独立的词法分析器从输入字符流中分离出一个个的“单词”，并将单词流作为其输入。实际开发中，语法分析器可以手工编写，也可以使用工具（半）自动生成。

（离散）什么是联结词完备集

设 S 是一个联结词集合，如果任何 n(n≥1)元真值函数都可以由仅含 S 中的联结词构成的公式表示，则称 S 是联

结词完备集

S={¬ ,∧,∨}是联结词完备集

（软工）软件开发流程

需求分析	1.相关系统分析员向用户初步了解需求，然后用 word 列出要开发的系统的大功能模块，每个大功能模块有哪些小功能模块，对于有些需求比较明确相关的界面时，在这一步里面可以初步定义好少量的界面。 2.系统分析员深入了解和分析需求，根据自己的经验和需求用 WORD 或相关的工具再做出一份文档系统的功能需求文档。这次的文档会清楚列出系统大致的大功能模块，大功能模块有哪些小功能模块，并且还列出相关的界面和界面功能。 3.系统分析员向用户再次确认需求。
概要设计	首先，开发者需要对软件系统进行概要设计，即**系统设计**。概要设计需要对软件系统的设计进行考虑，包括系统的基本处理流程、系统的组织结构、模块划分、功能分配、接口设计、运行设计、数据结构设计和出错处理设计等，为软件的详细设计提供基础。
详细设计	在概要设计的基础上，开发者需要进行软件系统的详细设计。在详细设计中，描述实现具体模块所涉及到的主要算法、数据结构、类的层次结构及调用关系，需要说明软件系统各个层次中的每一个程序(每个模块或子程序)的设计考虑，以便进行编码和测试。应当保证软件的需求完全分配给整个软件。详细设计应当足够详细，能够根据详细设计报告进行编码。
编码	在软件编码阶段，开发者根据《软件系统详细设计报告》中对数据结构、算法分析和模块实现等方面的设计要求，开始具体的编写程序工作，分别实现各模块的功能，从而实现对目标系统的功能、性能、接口、界面等方面的要求。在规范化的研发流程中，编码工作在整个项目流程里最多不会超过 1/2，通常在 1/3 的时间。
测试	测试编写好的系统。交给用户使用，用户使用后一个一个的确认每个功能。软件测试有很多种：按照测试执行方，可以分为内部测试和外部测试；按照测试范围，可以分为模块测试和整体联调；按照测试条件，可以分为正常操作情况测试和异常情况测试；按照测试的输入范围，可以分为全覆盖测试和抽样测试。以上都很好理解，不再解释。
软件交付	在软件测试证明软件达到要求后，软件开发者应向用户提交开发的目标安装程序、数据库的数据字典、《用户安装手册》、《用户使用指南》、需求报告、设计报告、测试报告等双方合同约定的产物。
验收	用户验收

（软工）什么是 UML

统一建模语言（UML，英语：Unified Modeling Language）是非专利的第三代建模和规约语言。UML 是一种开放的方法，用于说明、可视化、构建和编写一个正在开发的、面向对象的、软件密集系统的制品的开放方法。UML 展现了一系列最佳工程实践，这些最佳实践在对大规模，复杂系统进行建模方面，特别是在软件架构层次已经被验证有效。

（算法）阐述对一个数据集进行等价类划分的基本思想和过程

并查集应用:数据结构书 p267

（离散）简述欧拉回路的基本含义及一个应用。另，哈密顿回路

对于一个给定的连通图，怎样判断是否存在着一个恰好包含了所有的边，并且没有重复的路径？这就是一笔画问题。用图论的术语来说，就是判断这个图是否是一个能够遍历完所有的边而没有重复。这样的图现称为欧拉图。这时遍历的路径称作欧拉路径（一个环或者一条链），如果路径闭合（一个圈），则称为欧拉回路

一笔画问题

哈密顿图（英语：Hamiltonian path，或 Traceable path）是一个无向图，由天文学家哈密顿提出，由指定的起点前往指定的终点，途中经过所有其他节点且只经过一次。在图论中是指含有哈密顿回路的图，闭合的哈密顿路径称作哈密顿回路（Hamiltonian cycle），含有图中所有顶的路径称作哈密顿路径。

（？）简述对一个集合可能的表示方法（至少描述两种）
列举法，描述法..

（离散）简述逻辑命题与逻辑谓词的关系
？

（编译）简述用于定义语言句法的文法类型及其主要特征
乔姆斯基把方法分成四种类型，即 0 型、1 型、2 型和 3 型。
0 型，无限制文法。**无限制文法**是对文法的产生式左右两侧都没有限制的形式文法。
1 型，上下文相关文法。此文法对应于线性有界自动机。它是在 0 型文法的基础上每一个 α→β,都有|β|>=|α|。这里的|β|表示的是 β 的长度。另外允许 S→ε
2 型，上下文无关文法。它对应于下推自动机。2 型文法是在 1 型文法的基础上,再满足：每一个 α→β 都有 α 是非终结符。如 A->Ba,符合 2 型文法要求。
3 型，正规文法。它对应于有限状态自动机。它是在 2 型文法的基础上满足:A→α|αB（右线性）或 A→α|Bα（左线性）。

（编译）简述预测分析法 LL(1)的基本思想
自顶向下句法分析以文法开始符号为根，试图构造以输入符号串为树叶的推导树，每次都以最左边的那个非终结符为子树根，选择适当的产生式，往下生长子树。

Disadvantages of Operator-precedence Parsing

- handling tokens like the unary operators
- Since the relationship between a grammar and the operator-precedence parser is tenuous, one cannot always be sure the parser accepts exactly the desired language.
- Only a small class of grammars can be parsed using operator-precedence techniques.

Advantages of Operator-Precedence Parsing

- Simple, easy to construct by hand

预测分析表进行处理。

（编译）简述 LR(1)分析法的基本思想

LR(k) Parsing

- L: left-to-right scan
- R: construct a rightmost derivation in reverse
- k: the number of input symbols of look ahead

“向前看”，减少移近-规约冲突。

（编译）简述在程序流图中寻找循环的基本思想

- Input: A flow graph *G* and a back edge *n* -> *d*.
- Output: The set loop consisting of all nodes in the natural loop of *n* -> *d*.

注意 Back Edge

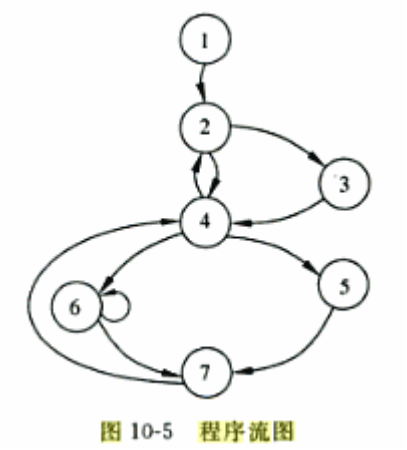


图 10-5　程序流图

```
Main( ){
   stack=empty;
   Loop={d};
   Insert(n);
   While stack is not empty {
      pop m, the first element of stack, off stack;
      for each predecessor p of m do
         insert(p);
    }
}
```

```
void insert (m){
   if m is not in loop {
      loop = loop ∪ {m};
      push m onto stack;
   }
}
```

（离散）举例偏序关系、全序关系，说明区别

偏序只对部分元素成立关系 R，全序对集合中任意两个元素都有关系 R。

例如：

集合的包含关系就是半序，也就是偏序，因为两个集合可以互不包含；

而实数中的大小关系是全序，两个实数必有一个大于等于另一个；

又如：复数中的大小就是半序，虚数不能比较大小。

知识面

（广）图灵机

图灵机，又称确定型图灵机，是英国数学家阿兰·图灵于 1936 年提出的一种抽象计算模型，其更抽象的意义为一种数学逻辑机，可以看作等价于任何有限逻辑数学过程的终极强大逻辑机器。

（离散）罗素悖论

（广）ACM 全称，是哪国的

计算机协会（Association of Computing Machinery，简称 ACM）是一个世界性的计算机从业员专业组织，创立于 1947 年，是世界上第一个科学性及教育性计算机学会。

（广）IEEE 全称

电气电子工程师学会（英语：Institute of Electrical and Electronics Engineers，简称为 IEEE，英文读作“i

triple e”[ai trɪpl i:]）是一个建立于 1963 年 1 月 1 日的国际性电子技术与电子工程师协会，亦是世界上最大的专业技术组织之一，拥有来自 175 个国家的 36 万会员。

除设立于美国纽约市的总部以外，亦在全球 150 多个国家拥有分会，并且还有 35 个专业学会及 2 个联合会。其每年均会发表多种杂志、学报、书籍，亦举办至少 300 次的专业会议。

（广）图灵奖及获得者

图灵奖（Turing Award，又译杜林奖），是计算机协会（ACM）于 1966 年设立的，又叫“A.M. 图灵奖”，专门奖励那些对计算机事业作出重要贡献的个人。其名称取自世界计算机科学的先驱、英国科学家，英国曼彻斯特大学教授艾伦·图灵（Professor Alan Turing），这个奖设立目的之一是纪念这位现代电脑、计算机奠基者。获奖者必须是在计算机领域具有持久而重大的先进性的技术贡献。大多数获奖者是计算机科学家。

（广）图灵测试

图灵测试（英语：Turing test，又译图灵试验）是图灵提出的一个关于判断机器是否能够思考的著名试验，测试某机器是否能表现出与人等价或无法区分的智能。
如果一个人（代号 C）使用测试对象皆理解的语言去询问两个他不能看见的对象任意一串问题。对象为：一个是正常思维的人（代号 B）、一个是机器（代号 A）。如果经过若干询问以后，C 不能得出实质的区别来分辨 A 与 B 的不同，则此机器 A 通过图灵测试。

（广）实时系统的分类

实时系统（Real-time operating system,RTOS）的正确性不仅依赖系统计算的逻辑结果，还依赖于产生这个结果的时间。实时系统能够在指定或者确定的时间内完成系统功能和外部或内部、同步或异步时间做出响应的系统。因此实时系统应该在事先定义的时间范围内识别和处理离散事件的能力；系统能够处理和储存控制系统所需要的大量数据。

强实时

强实时系统（Hard Real-Time）：在航空航天、军事、核工业等一些关键领域中，应用时间需求应能够得到完全满足，否则就造成如飞机失事等重大地安全事故，造成重大地生命财产损失和生态破坏。因此，在这类系统的设计和实现过程中，应采用各种分析、模拟及形式化验证方法对系统进行严格的检验，以保证在各种情况下应用的时间需求和功能需求都能够得到满足。

弱实时

弱实时系统（Soft Real-Time）：某些应用虽然提出了时间需求，但实时任务偶尔违反这种需求对系统的运行以及环境不会造成严重影响，如视频点播（Video-On-Demand，VOD）系统、信息采集与检索系统就是典型的弱实时系统。在 VOD 系统中，系统只需保证绝大多数情况下视频数据能够及时传输给用户即可，偶尔的数据传输延迟对用户不会造成很大影响，也不会造成像飞机失事一样严重的后果。

（广）并行技术有哪些

并行计算（英语：parallel computing）一般是指许多指令得以同时进行的计算模式。在同时进行的前提下，可以将计算的过程分解成小部份，之后以并发方式来加以解决。
相对于串行计算，并行计算可以划分成时间并行和空间并行。时间并行即流水线技术，空间并行使用多个处理器执行并发计算，当前研究的主要是空间的并行问题。以程序和算法设计人员的角度看，并行计算又可分为数据并行和任务并行。数据并行把大的任务化解成若干个相同的子任务，处理起来比任务并行简单。

空间上的并行导致两类并行机的产生，按照麦克·弗莱因（Michael Flynn）的说法分为单指令流多数据流（SIMD）和多指令流多数据流（MIMD），而常用的串行机也称为单指令流单数据流（SISD）。MIMD 类的机器又可分为常见的五类：并行矢量处理机（PVP）、对称多处理机（SMP）、大规模并行处理机（MPP）、工作站机群（COW）、分布式共享存储处理机（DSM）。

（广）什么是系列机

系列机指基本指令系统相同、基本体系结构相同的一系列不同型号的计算机。系列机的概念就是指先设计好一种系统结构，而后就按这种系统结构设计它的系统软件，按器件状况和硬件技术研究这种结构的各种实现方法。并按照

速度、价格等不同要求，分别提供不同速度、不同配置的各档机器。系列机必须保证用户看到的机器属性一致。

（广）多核处理器存在的问题

目前在多核技术的开发中需要解决 3 个重要问题：一是多核之间的竞争关系如何协调，二是多核的负载均衡如何实现，三是对如何实现对多核中内存、cache 等的管理。要解决这些问题，就需要软件开发和硬件开发在早期阶段进行合作，对某些功能进行定义。

（广）什么是 IP 电话

IP 电话（简称 VoIP，源自英语 Voice over Internet Protocol；又名宽带电话或网络电话）是一种透过互联网或其他使用 IP 技术的网络，来实现新型的电话通讯。过去 IP 电话主要应用在大型公司的内联网内，技术人员可以复用同一个网络提供数据及语音服务，除了简化管理，更可提高生产力。随着互联网日渐普及，以及跨境通讯数量大幅飙升，IP 电话亦被应用在长途电话业务上。由于世界各主要大城市的通信公司竞争日剧，以及各国电信相关法令松绑，IP 电话也开始应用于固网通信，其低通话成本、低建设成本、易扩充性及日渐优良化的通话质量等主要特点，被目前国际电信企业看成是传统电信业务的有力竞争者。