软微大数据技术介绍

2018年3月7日 9:24

大数据技术

主要研究大数据收集、交换、存储、分析、可视化的理论、方法和技术及相应的支撑平台，包括大数据机器学习的算法、模型与系统，大数据平台与云平台的可靠性、可用性和安全性增强技术和自动部署技术，面向智慧城市、医疗健康、人力资源等领域的大数据分析决策与精准服务。重点培养学生掌握扎实的数据科学基础理论和专业知识，具备从事大数据技术研究、数据平台构建、大数据分析和服务创新等工程实践活动的能力，成为适合大数据产业和技术发展要求的研究、开发与管理人才。

来自 <http://www.ss.pku.edu.cn/index.php/education/department/department-sedt>

大数据和机器学习的关系

2018年3月7日 9:46

现在已经成为大数据专业的研究生了，但是对于很多概念还是有点模糊，在网上查了一些资料，感觉略有心得，再次整理一下

大数据：大数据是相当于传统数据的概念，大数据的“大”体现在数据的 数量大，种类多，产生快，处理快，价值高等特点，大数据的学习路线又可以分为两种，一种是大数据 开发\分析\应用，以时下热门的hadoop和spark为主；另外一种是大数据的研发工作，也就是开发出大数据处理需要的数据库，统计平台，研发新的机器学习的算法等等。总之，分析是为了追求数据结果的，研发是为了更好的去分析。但是想在大数据领域学习下去，两种都必须有所涉及，但是学习要有侧重其中一种。

数据挖掘：从数据中提取潜在的、有价值的信息。这是一个比较宽泛的概念，使用机器学习算法来对大数据进行分析，找到有用的信息可以是数据挖掘，你从一张excel表中仔细观察，终于找到了几个有用的规律，这也算是数据挖掘。数据挖掘可以看成是对大数据处理的一种方式，但是大数据的处理方式并不止数据挖掘。

机器学习：机器学习简单的讲就是我们给计算机输入一些数据后，它必须做一些事情，也就是通过学习我们输入的数据，计算机要做出相应的反应，展示我们需要看到的结果。而且学习数据的过程是明确了，计算机通过我们设计的各种学习模式去学习数据，并通过从数据中学到的信息对计算机自身进行校正，以便于更好的学习，整个过程是迭代的。只要是采用了这种迭代并不断逼近的策略，一般都可以归到机器学习的范畴。要学习机器学习，各种学习模式是必须会的。之所以大数据和机器学习经常一起出现，是因为我们会使用机器学习这个工具做大数据的分析工作，也就是说机器学习可以看做是我们做大数据分析的一个比较好用的工具，但是大数据分析的工具并不止机器学习，机器学习也并不只能做大数据分析。

深度学习：深度学习就是一种比较火的机器学习算法，是基于神经网络发展起来的。

云计算：简而言之，就是将计算任务转移到服务器端，用户端只需要个显示器就可以。

来自 <http://blog.csdn.net/k_zer0/article/details/74360498>

数据库问题

2018年3月9日 21:46

？数据库插入的数据过多（数据库的数据量很大时）怎么办?

1.优化你的sql和索引
2.并不需要一定遵守范式理论，适度的冗余，让Query尽量减少Join
3.依据业务规则拆分成不同的细表

？视图与索引

- 视图
  - 视图的作用
    - 视图能够简化用户的操作
    - 视图使用户能以多种角度看待同一数据
    - 视图对重构数据库提供了一定程度的逻辑独立性
    - 视图能够对机密数据提供安全保护
    - 适当的利用视图可以更清晰的表达查询
- 索引
  - 建立索引
    - 唯一性索引
    - 聚簇索引（个张表最多一个，物理顺序按表顺序存放）
  - 删除索引
- 数据权限控制
  - 定义权限
  - 收回权限

？游标

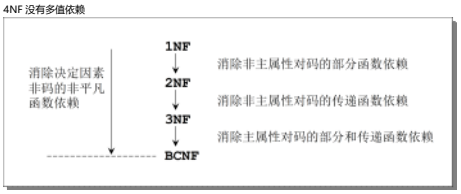
游标提供了在结果集中一次一行或者多行前进或向后浏览数据的能力。可以把游标当作一个指针，它可以指定结果中的任何位置，然后允许用户对指定位置的数据进行处理。
在Android中一般用于查询SQLite，也可用于查询ContentProvider，甚至可用于DownloadManager查询下载任务。
来自 <http://blog.csdn.net/aqi00/article/details/50423770>

？数据库的关系完整性

关系完整性的用于保证数据库中数据的正确性。系统在进行更新、插入或删除等操作时都要检查数据的完整性，核实其约束条件，即关系模型的完整性规则。
在关系模型中有四类完整性约束：实体完整性、域完整性、参照完整性和用户定义的完整性，其中实体完整性和参照完整性约束条件,称为关系的两个不变性。
1）实体完整性（Entity Integrity）：实体完整性指表中行的完整性。主要用于保证操作的数据（记录）非空、唯一且不重复。即实体完整性要求每个关系（表）有且仅有一个主键，每一个主键值必须唯一，而且不允许为“空”（NULL）或重复。
2）域完整性：（Domain Integrity）是指数据库表中的列必须满足某种特定的数据类型或约束。其中约束又包括取值范围、精度等规定
3）参照完整性（Referential Integrity）属于表间规则。对于永久关系的相关表，在更新、插入或删除记录时，如果只改其一，就会影响数据的完整性。如删除父表的某记录后，子表的相应记录未删除，致使这些记录称为孤立记录。对于更新、插入或删除表间数据的完整性，统称为参照完整性。
4）用户定义完整性（User-defined Integrity）是对数据表中字段属性的约束，用户定义完整性规则（User-defined integrity）也称域完整性规则。包括字段的值域、字段的类型和字段的有效规则（如小数位数）等约束，是由确定关系结构时所定义的字段的属性决定的。如，百分制成绩的取值范围在0~100之间等。

？范式

- 范式是符合某一种级别的关系模式的集合。
- 范式的种类：
  - 第一范式(1NF)
  - 第二范式(2NF)
  - 第三范式(3NF)
  - Boyce-Codd范式(BCNF)
  - 第四范式(4NF)
  - 第五范式(5NF)

- 1NF ⊃ 2NF ⊃ 3NF ⊃ BCNF ⊃ 4NF ⊃ 5NF
- 如果关系模式R的所有属性域都是原子（不可分的基本数据项）的，称R属于1NF,记为：R ∈ 1NF。
- 若关系模式R∈1NF，并且每一个非主属性都完全函数依赖于R的码，则R∈2NF。即表中其他数据元素都依赖于主关键字,或称该数据元素惟一地被主关键字所标识。第二范式是数据库规范化中所使用的一种正规形式。它的规则是要求数据表里的所有**非主属性都要和该数据表的主键有完全依赖关系；如果有哪些非主属性只和主键的一部份有关的话，它就不符合第二范式**
- 第三范式（3NF）3NF在2NF的基础之上，消除了非主属性对于码的传递函数依赖。也就是说，如果存在非主属性对于码的传递函数依赖，则不符合3NF的要求。
- **BCNF 在3NF的基础上消除主属性对于码的部分与传递函数依赖。**设关系模式R<U，F>∈1NF，如果对于R的每个函数依赖X→Y，若Y不属于X，则X必含有候选码，那么R∈BCNF。
- 4NF 没有多值依赖
- 

设有关系模式R(U)，U是属性集，X和Y是U的子集，如果X→Y是一个函数依赖，且对X的任何一个真子集X'都不存在X'→Y，则称X→Y是一个完全函数依赖(Full Functional Dependency)，称Y完全函数依赖于X。记为

$$X \xrightarrow{F} Y.$$

反之，如果X'→Y成立，则称X→Y是**部分函数依赖**(Partial Functional Dependency)，称Y部分函数依赖于X。记为 [2]

$$X \xrightarrow{P} Y.$$

图解表示

编辑

- 部分函数依赖可以用图1示意：属性集A由属性 k_1, k_2, k_3 构成(要求至少两个属性)，表示为 $A = \{k_1, k_2, k_3\}$ ；如果 $A \to k_4$ ，且在A中的一个属性 $k_2 \to k_4$ ，那么， $A \xrightarrow{P} k_4$ 。

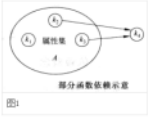


图1

码

设 K 为某表中的一个属性或属性组，若除 K 之外的所有属性都完全函数依赖于 K（这个“完全”不要漏了），那么我们称 K 为**候选码**，简称为**码。**在实际中我们通常可以理解为：**假如当 K 确定的情况下，该表除 K 之外的所有属性的值也就随之确定，那么 K 就是码。**一张表中可以有超过一个码。（实际应用中为了方便，通常选择其中的一个码作为**主码**）

作者：刘慰
链接：https://www.zhihu.com/question/24696366/answer/29189700
来源：知乎
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

给个简洁的判断方法：
1NF：每个数据项都是最小单元，不可分割，其实就是确定行列之后只能对应一个数据，形象点就是你不对Exel作拆分单元格。 其实数据库管理系统生成的最起码也是第一范式。
2NF：非主属性不部分依赖于候选码
3NF：非主属性不传递依赖于候选码
BCNF：在满足第二第三范式的情况下，主属性内部也不能部分或传递依赖。**判断方法**：箭头左边的必须是候选码，不是候选码的就不是BC范式。
4NF 没有多值依赖

作者：馨如姐姐
链接：https://www.zhihu.com/question/24696366/answer/49387513
来源：知乎
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

编程语言问题

2018年3月9日 21:57

Java

? 讲一讲Java反射机制的底层原理

http://www.fanyilun.me/2015/10/29/Java%E5%8F%8D%E5%B0%84%E5%8E%9F%E7%90%86/

Java的反射机制允许我们动态的调用某个对象的方法/构造函数、获取某个对象的属性等，而无需在编码时确定调用的对象，体现了多态的应用。有3种方法获取Class对象，类名.class, Class.forName(), 对象.getClass

http://blog.csdn.net/plgy_Y/article/details/72791483

? Java回收机制

但是任何一种垃圾回收算法一般要做两件基本事情：（1）发现无用的信息对象；（2）回收将无用对象占用的内存空间。使该空间可被程序再次使用。

判断一个对象是否可以被回收

1. 引用计数算法：判断对象的引用数量
2. 可达性分析算法：判断对象的引用链是否可达

垃圾收集算法

1、标记清除算法

标记-清除算法分为标记和清除两个阶段。该算法首先从根集合进行扫描，对存活的对象对象标记，标记完毕后，再扫描整个空间中未被标记的对象并进行回收

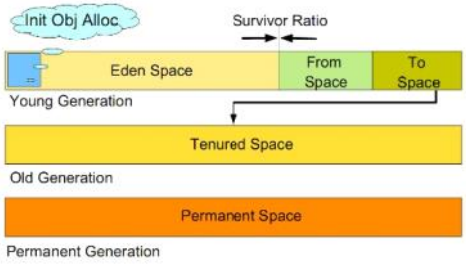
2、复制算法

复制算法将可用内存按容量划分为大小相等的两块，每次只使用其中的一块。当这一块的内存用完了，就将还存活着的对象复制到另外一块上面，然后再把已使用过的内存空间一次清理掉。

3、标记整理算法

标记整理算法的标记过程类似标记清除算法，但后续步骤不是直接对可回收对象进行清理，而是让所有存活的对象都向一端移动，然后直接清理掉端边界以外的内存，类似于磁盘整理的过程，该垃圾回收算法适用于对象存活率高的场景（老年代）

4、分代收集算法



1). 新生代（Young Generation）

新生代的目标就是尽可能快速的收集掉那些生命周期短的对象，一般情况下，所有新生成的对象首先都是放在新生代的。

2). 老年代（Old Generation）

老年代存放的都是一些生命周期较长的对象，就像上面所叙述的那样，在新生代中经历了N次垃圾回收后仍然存活的对象就会被放到老年代中。

3). 永久代（Permanent Generation）

永久代主要用于存放静态文件，如Java类、方法等。

C++

? C++程序内存区域

1. 自动存储类

- **自动存储变量存放在栈区**
- **进入声明的块时生成，在结束块时删除**
- **函数的参数和局部变量都是自动存储类**
- **自动存储是变量的默认状态**

2. 静态存储类

- **关键字extern和static声明静态存储类变量和函数标识符**
- **extern声明全局量（全局量默认为extern），static声明局部量**
- **extern和static说明变量时，程序开始执行时分配和初始化存储空间**
- **extern和static说明函数，表示从程序执行开始就存在这个函数名**

? Class中的成员性质

成员的性质由关键字public、protected、private 决定

public 公有 公有段的成员是提供给外部的接口

protected 保护 保护段成员在该类和它的派生类中可见

private 私有 私有段成员仅在类中可见

各段中既可以包含数据成员，也可以包含成员函数

? 构造函数的作用

为对象分配空间；对数据成员赋初值；请求其他资源

? 友元函数

- **友元是对类操作的辅助手段。友元能够引用类中本来被隐蔽的信息**
- **使用友元目的是基于程序的运行效率**
- **运算符重载的某些场合需要使用友元**
- **友元可以是函数，也可以是类**
- **友元关系是非传递的。即 Y 是 X 的友元，Z 是 Y 的友元，但 Z 不一定是X的友元**

友元提供了一种 普通函数或者类成员函数 访问另一个类中的私有或保护成员 的机制。也就是说有两种形式的友元：

（1）友元函数：普通函数对一个访问某个类中的私有或保护成员。

（2）友元类：类A中的成员函数访问类B中的私有或保护成员。

2.特性

优点：提高了程序的运行效率。

缺点：破坏了类的封装性和数据的透明性。

? 类之间的关系

has-A，uses-A 和 is-A

has-A **包含关系，用以描述一个类由多个“部件类”构成。实现has-A关系用类成员表示，即一个类中的数据成员是另一种已经定义的类。**

uses-A **一个类部分地使用另一个类。通过类之间成员函数的相互联系，定义友元或对象参数传递实现。**

is-A **表示一种分类方式,描述类的抽象和层次关系,该机制称为“继承”。is - a 关系具有传递性,不具有对称性。**

? 什么是多态，怎么实现

我自己理解的多态是程序在运行期间，依据所指的对象，动态地调用其重载的实现方法。

C

? static 全局 局部变量分别在哪个存储区

static全局变量只在声明此static全局变量的文件中有效；
普通全局变量对整个源程序都有效，当此源程序包含多于一个文件的程序时，对其他文件依然有效。
https://www.cnblogs.com/bakari/archive/2012/08/05/2623637.html
https://www.cnblogs.com/wuyuankun/p/3675996.html

? const void *a 与 void *const a 的区别

const void *a
这是定义了一个指针a，a可以指向任意类型的值，但它指向的值必须是常量。
在这种情况下，我们不能修改被指向的对象，但可以使指针指向其他对象。
例如：
const void *a；*a=0x123;//是编译通不过的，因为*a中放的是个const值。const值是不能被改变的。
const int m=1;   const int n=2;
a=&m;   a=&n;//编译可以通过
void* const a
这是定义了一个const指针a，a可以指向任意类型的值，但a是指向某个对象的常量指针。
我们不能修改指针中存储的地址，但可以修改指针指向的对象。
例如：
void* const a;这个定义：*a=0x123;是没问题的，
但是a=(void*)&b;是不行的，因为a是const变量。
如：
int m=1;   nt n=2;
a=&m;   a=&n;//编译不成功。

可以这么说，const void *a；中const修饰的是*a。在void* const a中，const 修饰的是a。

? static关键字

当用于不同的上下文环境时，static关键字具有不同的意义。

当它用于函数定义时，或用于代码块之外的变量声明时，static关键字用于修改标示符的连接属性，从external改为internal，但标示符的存储类型和作用域不受影响。用这种方式声明的函数或变量只能在它们的源文件中访问。

当它用于代码块内部的变量声明时，static关键字用于修改变量的存储类型，从自动变量改为静态变量，但变量的连接属性和作用域不受影响。用这种方式声明的变量在程序执行之前创建，并在程序的整个执行期间一直存在，而不是每次在代码块开始执行是创建，在代码块执行完毕后销毁

联编是指一个程序模块、代码之间互相关联的过程。
静态联编，是程序的匹配、连接在编译阶段实现，也称为早期匹配。
  重载函数使用静态联编。
动态联编是指程序联编推迟到运行时进行，所以又称为晚期联编。
  switch 语句和 if 语句是动态联编的例子。
冠以关键字 virtual 的成员函数称为虚函数
实现运行时多态的关键首先是要说明虚函数，另外，必须用
基类指针调用派生类的不同实现版本

- **一个虚函数，在派生类层界面相同的重载函数都保持虚特性**
- **虚函数必须是类的成员函数**
- **不能将友元说明为虚函数，但虚函数可以是另一个类的友元**
- **析构函数可以是虚函数，但构造函数不能是虚函数**
- **纯虚函数是一个在基类中说明的虚函数，在基类中没有定义，要求任何派生类都定义自己的版本**
- **纯虚函数为各派生类提供一个公共界面**
- **纯虚函数说明形式：**
  virtual *类型 函数名（参数表）* = 0；
- **一个具有纯虚函数的基类称为抽象类。**

指针和引用的联系与区别

★ 相同点：
1. 都是地址的概念；
指针指向一块内存，它的内容是所指内存的地址；引用是某块内存的别名。
★ 区别：
1. 指针是一个实体，而引用仅是个别名；
2. 引用使用时无需解引用(*)，指针需要解引用；
3. 引用只能在定义时被初始化一次，之后不可变；指针可变；
4. 引用没有 const，指针有 const；
5. 引用不能为空，指针可以为空；
6. “sizeof 引用”得到的是所指向的变量(对象)的大小，而“sizeof 指针”得到的是指针本身(所指向的变量或对象的地址)的大小；
7. 指针和引用的自增(++)运算意义不一样；
8.从内存分配上看：程序为指针变量分配内存区域，而引用不需要分配内存区域。

计算机网络问题

2018年3月9日 21:59

SSL和TLS

SSL(Secure Sockets Layer) 1.0 2.0 3.0 是网景公司设计，google在2014发现设计缺陷，人们在其基础上提出改进协议就叫TLS(Transport Layer Security)，最新版本是TLS1.2，1.3还处理草案过程中。

计算机网络各层的协议

7	**应用层** application layer	例如HTTP、SMTP、SNMP、FTP、Telnet、SIP、SSH、NFS、RTSP、XMPP、Whois、ENRP
6	**表示层** presentation layer	例如XDR、ASN.1、SMB、AFP、NCP
5	**会话层** session layer	例如ASAP、SSH、ISO 8327 / CCITT X.225、RPC、NetBIOS、ASP、IGMP、Winsock、BSD sockets
4	**传输层** transport layer	例如TCP、UDP、TLS、RTP、SCTP、SPX、ATP、IL
3	**网络层** network layer	例如IP、ICMP、IPX、BGP、OSPF、RIP、IGRP、EIGRP、ARP、RARP、X.25
2	**数据链路层** data link layer	例如以太网、令牌环、HDLC、帧中继、ISDN、ATM、IEEE 802.11、FDDI、PPP
1	**物理层** physical layer	例如线路、无线电、光纤

屏幕剪辑的捕获时间: 2018/3/10 16:20

注RIP是应用层协议，采用TCP，端口号520

常见应用层协议总结

应用程序	FTP数据链接	FTP控制链接	TELNET	SMTP	DNS	TFTP Trivial File Transfer Protocol	HTTP	POP3	SNMP（简单网络管理协议）	RIP	DHCP	BGP	IMAP(Internet Message Access Protocol)	LDAP(用于单点登陆) Lightweight Directory Access Protocol
使用协议	TCP	TCP	TCP	TCP	UDP	UDP	TCP	TCP	TCP	UDP	UDP	TCP	TCP	both
端口号	20	21	23	25	53	69	80	110	161	520	67发,68收	179	143, IMAP over SSL (IMAPS) 993	389，636（SSL）

IPsec

In computing, Internet Protocol Security (IPsec) is a network protocol suite that authenticates and encrypts the packets of d ata sent over a network. IPsec includes protocols for establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to use during the session.

Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network.（应用层）

第二层隧道协议（英语：Layer Two Tunneling Protocol，缩写为L2TP)是一种虚拟隧道协议，通常用于虚拟专用网。L2TP协议自身不提供加密与可靠性验证的功能，可以和安全协议搭配使用，从而实现数据的加密传输。经常与L2TP协议搭配的加密协议是IPsec，当这两个协议搭配使用时，通常合称L2TP/IPsec。

1.RIP v2和BGP是应用程，理由：使用TCP/UDP端口
2.EIGRP属于传输层，理由：使用协议号
3.OSPF属于网络层，理由：直接封装在2层协议之上

UDP怎么确保可靠

简单来讲，要使用UDP来构建可靠的面向连接的数据传输，就要实现类似于TCP协议的
1. 超时重传（定时器）
2. 有序接受 （添加包序号）
3. 应答确认 （Seq/Ack应答机制）
4. 滑动窗口流量控制等机制 （滑动窗口协议）

目前已经有一些实现UDP可靠传输的机制，比如UDT（UDP-based Data Transfer Protocol）基于UDP的数据传输协议（UDP-based Data Transfer Protocol，简称UDT）是一种互联网数据传输协议。

IP的作用？IPv4和IPv6的联系

IP使用相互异构的网络在使用IP协议后进行相互通信，当联网网上的主机通信时，就好像在一个网络上通信一样，屏蔽了各种异构的细节。IPv6是IPv4的升级，主要是全球IPv4地址已经枯竭，无论是采用CIDR还是NAT转换都不能从根本上解决IPv4地址枯竭的情况，IPv6主要是1）更大的地址空间 2）灵活的首部格式 3)支持即插即用 4）只有在包的源能分片 ，是端到端的，传输路径中的路由不能分片 5）IPv6首部长度必须是8B整数部，而IPv4首部是4B的整数倍6）增加了安全性，有身份验证和保密功能
IPv6数据报的目的地址有三种基本类型

1. 单播
2. 多播
3. 任播

有连接和无连接协议的区别

面向连接服务:通信前双方必须先建立连接,分配相应的资源(如缓冲区),以保存通信能正常进行,传输结束后释放连接和所占用的资源.因此这种服可以分为连接建立、数据传输和连接释放这三个阶段。TCP就是一种面向连接服务的协议。
无连接服务：通信双方不需要先建立连接，需要发送数据时就直接发送，把每个带有目的地址的包传送到线路上，由系统选定路线进行传输，是一种不可靠的服务。这种服务常被描述为“尽最大努力交付”，并不保证通信的可靠。如 IP、UDP就是一种无连接的服务协议。

可靠服务和不可靠服务

可靠服务：指网络具有纠错、检错、应答机制，能保证数据正确、可靠地传送到目的地。
不可靠服务：指网络只是尽量正确、可靠地传送，但不能保证数据正确、可靠地传送到目的地，是一种尽力而为的服务。

有应答服务和无应答服务

有应答服务：指接收方在收到数据后向发送方给出相应的应答，该应答由传输系统内部自动实现，而不是由用户实现。所发送的应答可以是肯定应答，也可以是否定应答，通常在接收到的数据有错误时发送否定应答。例如，文件传输服务就是一种有应答服务。
无应用服务：指接收方收到数据后不自动给出应答。若需要应答，则由高层实现。例如WWW服务，客户端收到服务器发送的页面文件后不给出应答。

C/S与B/S

C/S是双向通讯 B/S是查询式通讯
c/s是客户机(client)/服务器（server），b/s是浏览器（browser)/服务器。c/s之间通过任意的协议通信，一般要求有特定的客户端。比如QQ就是c/s模式，你的桌面上的QQ就是腾讯公司的特定的客户端，而服务器就是腾讯的服务器。再比 如你看的网络电视也是如此，比如你的桌面上的pplive、Tvcoo等，这些软件都是c/s模式的，他们要求在用户有特定的客户端。而B/S模式是靠应用层的http协议进行通信的（当然也要靠底层的好多协议支持），一般不需要特定的客户端，而是需要有统一规范的客户端，那就是你的浏览器！Web页就是B/S 模式，也就是说咱们说的网站就是B/S模式。

? **中继系统**

网线层以上	网关
网络层	路由器
数据链路层	网桥：工作在链接层的MAC子层（1.透明网桥 2.源路由网桥（最佳路由）） 局域网交换机：实质是多端口网桥
物理层	中继器：信号放大再转化，连接的是网段而不是子网 集线器：实质是多端口中继器

? 网卡的几中工作模式

Host-only模式

只有主机能够访问虚拟机，同时该模式下的网卡也不能访问外部机器，但是虚拟机之间可以互相通信。

Bridged模式

桥接模式，将虚拟机网卡桥接到物理网卡。所以采用桥接模式的虚拟机能够和主机所在的局域网中的PC进行通信。其他局域网的主机也能访问到虚拟主机。

NAT模式

地址转换模式。该模式下虚拟机能访问外部主机局域网中的机器，但是局域网中的机器不能访问虚拟机。

数据结构与算法问题

2018年3月9日 21:59

? 顺序存储和链式存储的区别

顺序表的特点是逻辑上相邻的数据元素，物理存储位置也相邻，并且，顺序表的存储空间需要预先分配。在链表中逻辑上相邻的数据元素，物理存储位置不一定相邻，它使用指针实现元素之间的逻辑关系。并且，链表的存储空间是动态分配的。两种存储结构各有长短，选择哪一种由实际问题中的主要因素决定。通常“较稳定”的线性表，即主要操作是查找操作的线性表，适于选择顺序存储；而频繁做插入删除运算的（即动态性比较强）的线性表适宜选择链式存储。

? 爬楼梯（斐波那契数）一共有n个台阶，你一次可以走一个台阶，或者两个台阶。那么，走到台阶顶时，一共有多少种走法。

直接思路无论是递归还是迭代法

递归法：时间复杂度：O(n)，空间复杂度：O(n)

迭代法：时间复杂度：O(n)，空间复杂度：O(1)

f(n) = f(n-1) + f(n-2)

用二阶矩阵相乘可以得到，算法复杂度到O(log(n))

? 如何判断单链表是否存在环

给定一个单链表，只给出头指针h：

1、如何判断是否存在环?

2、如何知道环的长度?

3、如何找出环的连接点在哪里?

4、带环链表的长度是多少?

解法：

1、对于问题1，使用追赶的方法，设定两个指针slow、fast，从头指针开始，每次分别前进1步、2步。如存在环，则两者相遇；如不存在环，fast遇到NULL退出。

2、对于问题2，记录下问题1的碰撞点p，slow、fast从该点开始，再次碰撞所走过的操作数就是环的长度s。

3、问题3：有定理：碰撞点p到连接点的距离=头指针到连接点的距离，因此，分别从碰撞点、头指针开始走，相遇的那个点就是连接点。

该定理的证明可参考：http://fayaa.com/tiku/view/7/

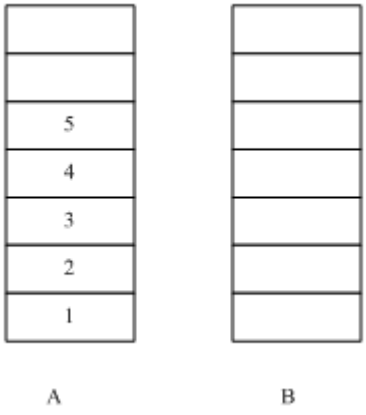
4、问题3中已经求出连接点距离头指针的长度，加上问题2中求出的环的长度，二者之和就是带环单链表的长度

? 双栈模拟队列

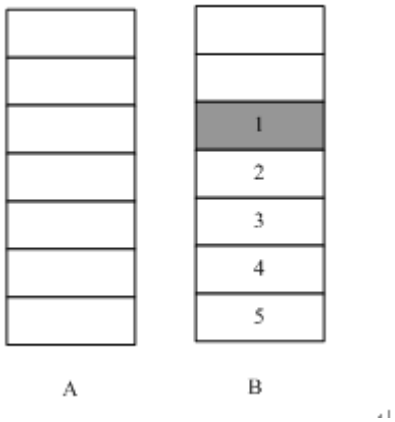
双栈模拟队列的实现：

初始时两个栈 A 与栈 B 都为空

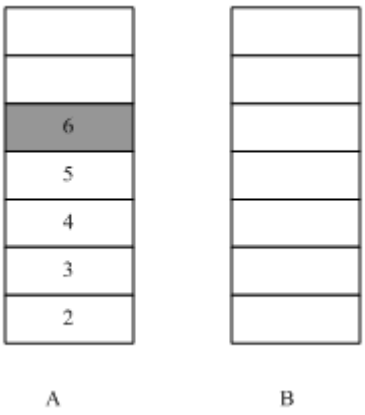
a. 入队： 1 2 3 4 5 顺序压入栈 A 中，栈 B 为空。



b. 出队：先将栈 A 的元素弹到栈 B 中，将栈 B 的顶部数据 1 出栈，即完成出队操作



c. 再次入队：先将栈 B 中的数据弹出到栈 A 中，然后将数据 6 压入栈 A 的顶部，即完成入队操作



来自 <http://blog.csdn.net/summersunboy/article/details/6138740>

操作系统问题

2018年3月9日 21:59

操作系统特征

1.并发

2.共享

3.虚拟：虚拟处理器（多道程序设计技术）、虚拟存储器、 虚拟设备

4.异步

进程为什么是资源分配的基本单位

进程作为资源分配基本单位，而把线程引入后作为处理机分配单元，可以提高系统并发度，提高系统吞吐量。

进程是具有一定独立功能的程序关于某个数据集合上的一次运行活动,进程是系统进行资源分配和调度的一个独立单位.

线程是进程的一个实体,是CPU调度和分派的基本单位,它是比进程更小的能独立运行的基本单位.同一个进程中的线程主要是完成进程独立功能的不同方面，需要的资源统一按进程分配有利于避免把资源按线程分配后竞争后产生的互斥，且这种情况下同一进程的线程切换所带来的时空开销非常大，因此进程是资源分配的基本单位。

进程与线程的区别

引入进程，是为了更好地使多道程序并发执行，以提高资源利用率和系统吞吐量，增加并发程度；而引入线程，是为了减小程序在并发执行时所付出的时空开销，提高操作系统的并发性能。在引入线程后，进程只作为除CPU以外系统资源的分配单元，线程则作为处理机的分配单元。

线程与进程的比较

(1) 调度。在传统的操作，拥有资源和独立调度的基本单位都是进程。在引入线程的操作系统中，线程是独立调度的基本单位，进程是资源拥有的基本单位。在同一进程中，线程的切换不会引起进程切换。在不同的进程中进行线程切换，会引起进程切换。

(2) 拥有资源。不论是传统操作系统还是设有线程的操作系统，进程都是拥有资源的基本单位，而线程不拥有系统资源（也有一点必不可少的资源），但线程可以访问其隶属的进程的系统资源。

(3) 并发性。在引入线程的操作系统中，不仅进程之间可以并发执行，而且多个线程之间也可以并发执行，从而使操作系统具有更好的并发性，提高了系统的吞吐量。

(4) 系统开销。由于创建或撤销进程时，系统都要为之分配或回收资源，如内存空间、I/O设备等，因此操作系统所付出的开销远大于创建或撤销线程时的开销。类似地，在进行进程切换时，涉及当前执行进程CPU环境的保存及新调度到进程CPU环境的设置，而线程切换时只需保存和设置少量寄存器内容，开销很小此外，由于同一进程内的多个线程共享进程的地址空间，因此，这些线程之间的同步与通信非常容易实现，甚至无需操作系统的干预。

(5) 地址空间和其他资源（如打开的文件）：进程的地址空间之间相互独立，同一进程的各线程共享进程的资源，某进程内的线程对于其他线程不可见。

(6) 通信方面：进程间通信（IPC）需要进程同步和互斥手段的辅助，以保证下数据的一致性，而线程间可以直接读/写进程数据段（如全局变量）来进行通信。

什么是死锁，死锁产生的原因，发生死锁的必要条件，死锁的处理

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

死锁生产的原因

系统资源的竞争。对系统中不可剥夺的资源竞争而陷入僵局

进程推进顺序非法。进程运行过程中，请求和释放资源的顺序不当，也同样会导致死锁。

生产死锁的必要条件

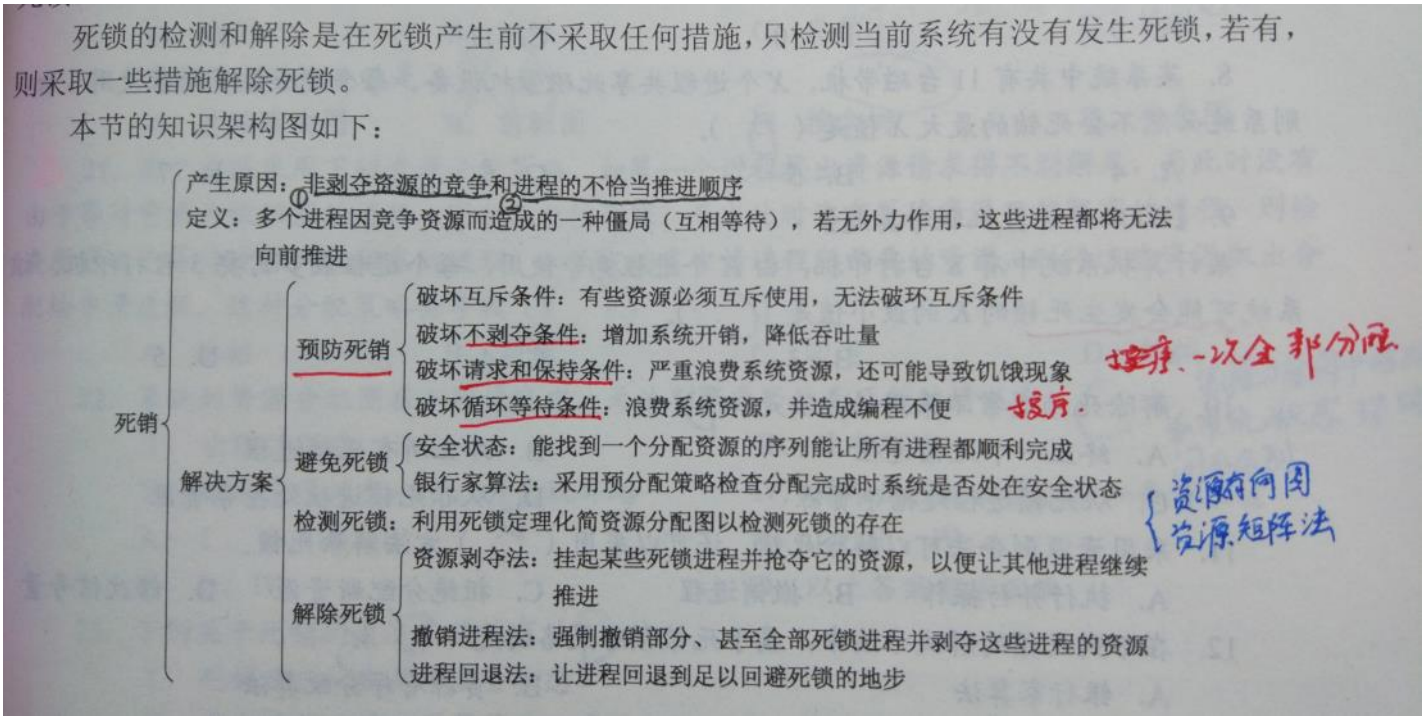
互斥条件

不剥夺条件

请求和保持条件

循环等待条件

死锁的处理策略



? 微内核vs宏内核

微内核（Monolithic Kernel）、宏内核（Micro Kernel）

微内核 是将各种服务功能放到内核之外,自身仅仅是一个消息中转战,用于各种功能间的通讯.

宏内核 是将所有服务功能集成于一身,使用时直接调用.

微内核的系统有WindowNT,Minix,Mach,etc.

宏内核的系统有Unix,Linux,etc.

? 内存分配策略

连续分配

1. 单一连续分配
2. 固定分区分配
3. 动态分区分配
    a. 首次适应
    b. 最佳适应
    c. 最坏适应
    d. 邻近适应

非连续分配

1. 基本分页
2. 基本分段
3. 段页式

UML问题

2018年3月9日　21:59

? 面向对象建模

用面向对象方法开发软件，通常需要建立三种形式的模型，它们分别是：

（1）对象模型：描述系统的数据结构；

（2）动态模型：描述系统的控制结构；

（3）功能模型：描述系统的功能。

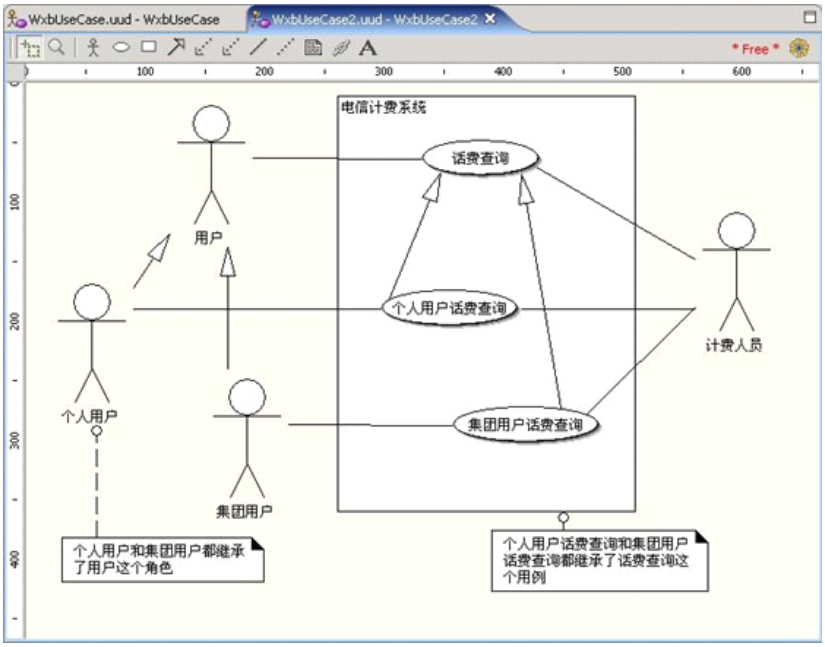
3种模型必不可少，其重要程度不同，对象模型是最基本、最重要的。

UML

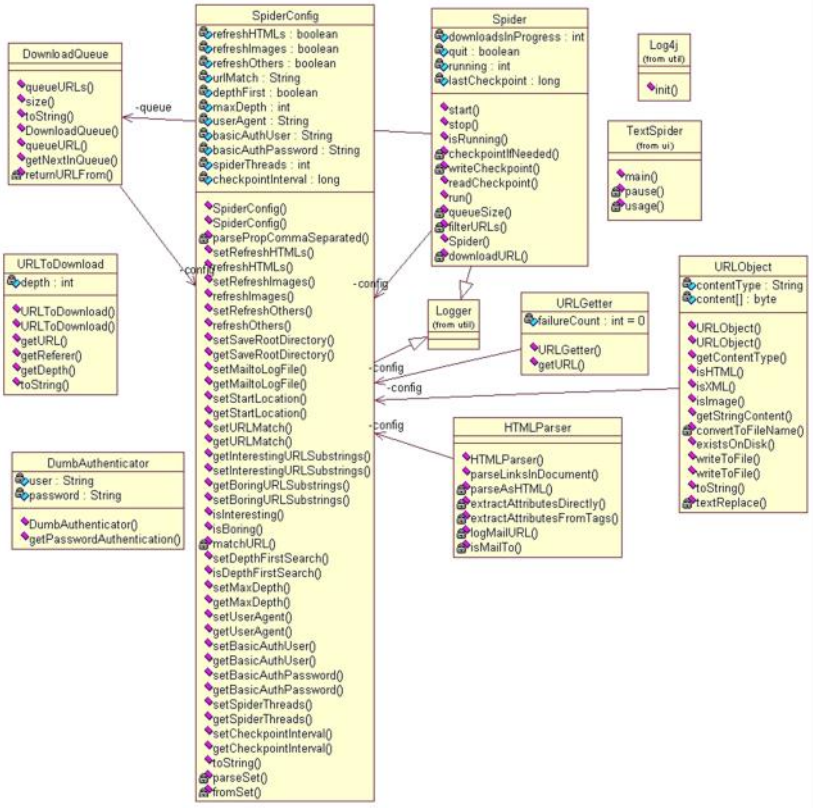
- **UML的定义包括UML语义和UML表示法两部分。**

  UML的重要内容可以由五类图（9种图形）来定义：

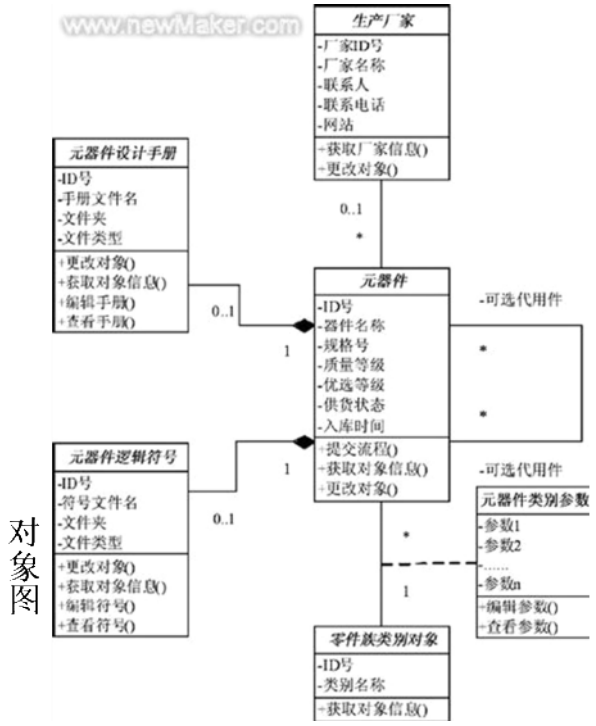
  1）用例图：从用户角度描述系统功能，并指出各功能



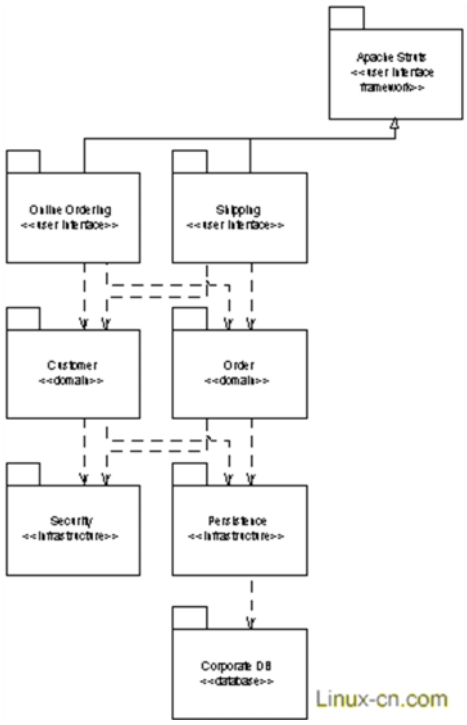
  2）静态图：包括类图、对象图和包图；



类图



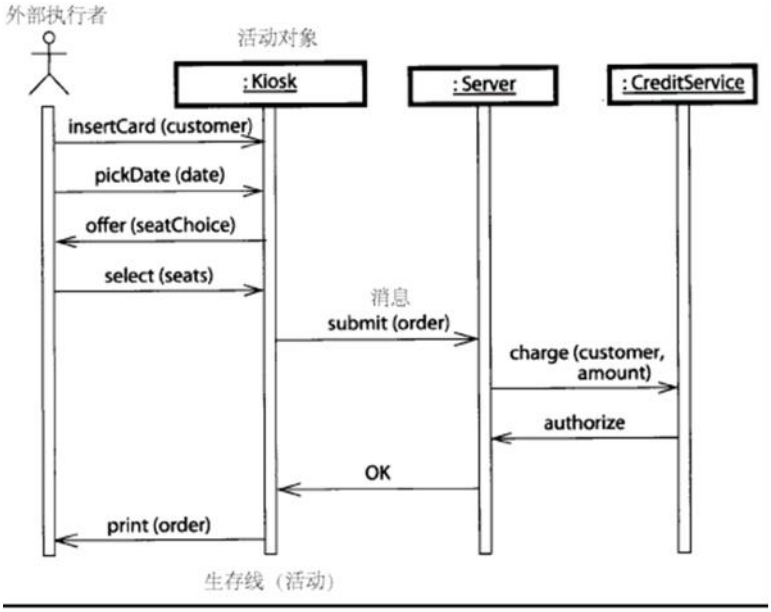
对象图



包图

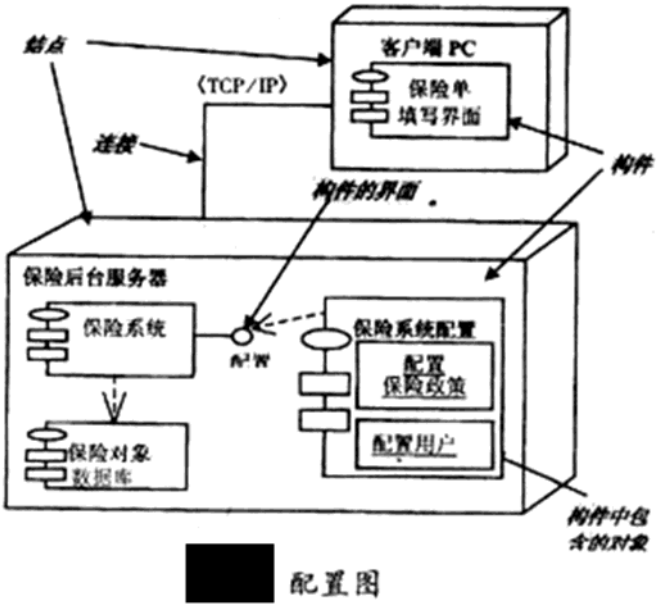
3）行为图：描述系统的动态模型；

4）交互图：描述对象间的交互关系；



交互图

5）实现图：如配置图，定义系统中软硬件的物理体系结构。



配置图

软件工程问题

2018年3月9日 21:59

? 软件生命周期各个阶段

1.问题定义；
2.可行性研究；
3.需求分析；
4.总体设计（概要设计）；
5.详细设计；
6.编码与单元测试；
7.综合测试；
8.维护。

? 耦合与内聚

耦合：指软件结构内**不同模块**彼此之间相互依赖（连接）的紧密程度。 模块耦合分为4类，程度不断加深

1. 数据耦合
2. 控制耦合
3. 公用耦合
4. 内容耦合

内聚：一个**模块内**部各个元素彼此结合的紧密程度。

它是衡量一个模块内部组成部分间整体统一性的度量。下述的内聚逐渐降低

1. 功能内聚
2. 顺序内聚
3. 通信内聚
4. 过程内聚
5. 时间内聚
6. 逻辑内聚
7. 偶然内聚

? 软件测试**方法**

黑盒测试

如果已经知道软件应该具有的功能，可以通过测试来检验是否每个功能都能正常使用，这种测试称黑盒测试。也称功能测试。

白盒测试

也称结构测试。

如果知道软件内部工作过程，可以通过测试来检验软件内部动作是否按照规格说明书的规定正常进行，这种测试称为白盒测试。

? 软件测试的**步骤**

1.模块测试

模块测试又称**单元测试**，它把每个模块作为单独的实体来测试。

2.子系统测试

子系统测试是把经过单元测试的模块放在一起形成一个子系统来测试。

3.系统测试

系统测试是把经过测试的子系统装配成一个完整的系统来测试。

4.验收测试

验收测试把软件系统作为单一的实体进行测试（利用用户的实际数据测试）。

5.平行运行

平行运行是同时运行新开发出来的系统和将被它取代的旧系统，以便比较新旧两个系统的处理结果。

集成测试

集成测试是组装软件的系统化技术，它将经过单元测试的模块联系在一起进行测试。

自顶向下集成

自底向上集成

回归测试

指重新执行已经做过的部分测试。

回归测试用于保证由于调试或其他原因引起的程序变化，不会导致额外错误的测试活动。

确认测试（验收测试）

也称为验收测试，目标是验证软件的有效性。

如果软件的功能和性能符合用户的期待，软件就是有效的。

? 面向对象基本概念

1、类（Class）

类就是对具有相同数据和相同操作的一组相似对象的定义。

2、实例（Instance）

实例就是由某个特定的类所描述的一个具体的对象。

3、消息（Message）

消息就是用来请求对象执行某个处理或回答某些信息的要求。

一个消息由三个部分组成：

1）接收信息的对象；

2）信息选择符（即消息名）；

3）零个或多个变元（参数）。

如：MyCircle.Show(GREEN);

4、方法（Method）

方法，是对象所能执行的操作。

C++中把方法称为成员函数，如Circle类中定义的成员函数：Show(int color)

5、属性（Attribute）

属性，是类中定义的数据。

C++中把属性称为数据成员。

6、封装（encapsulation）

封装就是信息隐藏，通过封装对外界隐藏了对象的实现细节。

7、继承（Inheritance）

继承，是指能够直接获得已有的性质和特征，而不必重复定义它们。

8、多态性（Polymorphism）

多态性，指子类对象可以象父类对象那样使用，同样的消息既可以发送给父类对象，也可以发送给子类对象。

即不同等级的类，可以公用一个方法的名字。

C++中，多态性是通过虚函数来实现的。在不同层次的类中，虚函数实现算法不同，在运行时根据接收消息的对象所属于的类来决定执行虚函数的版本，称为动态联编。

9、重载（Overloading）

有两种重载：

1）函数重载

2）运算符重载

? 软件开发模型

瀑布模型 waterful model

优点：采用规范的结构化方法；严格规定每个阶段提交的文档；要求每个阶段交出的产品必须经过验证

缺点：对如何处理开发中产品和活动的变化没有提供相关指导；将软件开发视为制造而不是创造；创造一个产品没有迭代的活动；需要等待很长的时间

快速原型模型 fast-prototyping model

优点：克服瀑布模型的缺点，减少由于软件需求不明确带来的开发风险。

这种模型适合预先不能确切定义需求的软件系统的开发。

缺点：所选用的开发技术和工具不一定符合主流的发展；快速建立起来的系统结构加上连续的修改可能会导致产品质量低下。

使用这个模型的前提是要有一个展示性的产品原型，因此在一定程度上可能会限制开发人员的创新。

增量模型 incremental model

优点：

短时间内可提交完成部分功能的产品。

逐渐增加产品功能，用户适应产品快。

缺点：

增量构件划分以及集成困难。

螺旋模型 spiral model

优点：有利于软件重用，重视软件质量；减少过多测试

缺点：风险驱动，需要丰富的风险评估经验；主要适用于内部开发的大规模软件项目；
随着迭代次数增加，工作量加大，开发成本增加

喷泉模型 fountain model

软件开发过程自下而上周期的各阶段是相互迭代和无间隙的特性
1、喷泉模型的优点
喷泉模型不像瀑布模型那样，需要分析活动结束后才开始设计活动，设计活动结束后才开始编码活动。该模型的各个阶段没有明显的界限，开发人员可以同步进行开发。其优点是可以提高软件项目开发效率，节省开发时间，适应于面向对象的软件开发过程。
2、喷泉模型的缺点
由于喷泉模型在各个开发阶段是重叠的，因此在开发过程中需要大量的开发人员，因此不利于项目的管理。此外这种模型要求严格管理文档，使得审核的难度加大，尤其是面对可能随时加入各种信息、需求与资料的情况。

敏捷开发方法和极限编程的特点 agile model

（1）敏捷方法
强调灵活性在快速、有效开发在软件中的作用
相对于过程和工具，更强调个人和交互的价值
更喜欢在生产运行的软件上投入时间，而不是在文档的编写上
注重客户的合作，而不是合同谈判
专注于对变化的反应，而不是创建一个计划而后遵循这个计划
（2）极限编程
具有强沟通、简化设计和迅速反馈等特点，一般只适合于规模小、进度紧、需求不稳定、开发小项目的小团队。极限编程的核心有4个要点：交流、简单、反馈和勇气。

? 简述自顶向下和自底向上两种集成测试方法

自顶向下的集成是从主控模块（主程序，即根结点）开始，按照系统程序结构，沿着控制层次从上而下，逐渐将各模块组装起来。在从上向下的集成测试过程中，需对那些未经集成的模块开发桩模块。在集成过程中，可以采用宽度优先或深度优先的策略向下推进。

自底向上的集成是从最底层模块（即叶子结点）开始，按照调用图的结构，从下而上，逐层将各模块组装起来。在从下而上的集成测试环境中，需对那些未经集成测试的模块开发驱动模块。

组成原理问题

2018年3月9日 21:59

? 透明在计算机的含义,哪些符合透明特征

客观存在并且运行着但是我们看不到的特性，简单讲，透明就是黑盒，只需要应用给出的接口，不需要了解内在的机理。

联想存储器TLB， 虚拟内存，在CPU中，IR、MAR、和MDR对各类程序员都是透明的，网络中的透明传输，

? 什么是虚拟内存，或虚拟内存由哪些组成

主存和联机工作的辅存共同构成了虚拟存储器，二者在硬件和系统软件的共同管理下工作。对于应用程序员而言，虚拟存储器是透明的。虚拟存储器将主存或辅存的地址空间统一编址，形成一个庞大的地址空间，在这个空间内，用户可以自由编程，而不必在乎实际的主存空间和程序在主存中实际的存放位置。

? 影响流水线的因素

1.结构相关（资源冲突）

由于多条指令在同一时刻急用一同资源而形成的冲突称为结构相关，有以下两种解决办法：

(1) 在前一指令访存时，使后一条相关指令（以及其后续指令）暂停一个时钟周期。

(2) 单独设置数据存储器和指令存储器，使两项操作条自在不同的存储器中进行，这属于资源重复配置。

2.数据相关（数据冲突）

存在必须等前一条指令执行完才能执行后一条指令的情况，则这两条指令即为数据相关。当多条指令重叠处理时就会发生冲突，解决的办法有以下两种：

(1) 把遇到的数据相关指令及其后续的指令都暂停一至几个时钟周期，直到数据相关问题消失后再继续执行。

(2) 设置相关的专用通路（旁路），即不等前一条指令把计算结果写回寄存器组，下一条指令也不再读寄存器组，而是直接把前一条指令的ALU的计算结果作为自己的输入数据开始计算过程，使需要暂停的操作变得可以继续执行，称为数据旁路技术。

3.控制相关（控制冲突）

当流水线遇到转移指令和其他改变PC值的指令而造成断流时，会引起控制相关。解决的办法有以下几种：

(1) 尽早差别转移是否发生，尽早生成转移目标地址。

(2) 预取转移成功和不不成功两个控制流方向上的目标指令。

(3) 加快和提前形成条件码。

(4) 提高转移方向的猜准率。

? 超标题流水线概念

1.超标题流水线技术

每个时钟周期内可并发多条独立指令，需要配置多个功能部件。

2.超流水线技术

在一个时钟周期内再分段，在一个时钟周期内一个功能部件使用多次。

3.超长指令字

由编译程序挖掘出指令间潜在的并行性，将多条能并行操作的指令组成一条具有多少操作码字段的越长指令字（可达几百位），需要采用多个处理部件。

? I/O方式有哪些

有程序查询、程序中断、DMA和通道

程序查询

由CPU不断查询程序运行的状态，接口设计简单，设备少，但CPU很多时间处理忙等，效率非常低，实时性好，在一些导弹路径计算等实时性要求高的地方仍然有使用

程序中断

在计算机执行程序过程中，出现紧急处理的异常或特殊情况时，CPU暂时中止现行程序，而转去这些情况，并在处理完毕后自动返回现行程序的断点继续执行

- **外中断**：来自处理器和内存以外的部件引起的中断，包括I/O设备发出的I/O中断、外部信号中断、以及各种定时器中断等
- **内中断**：处理器和内存内部产生的中断，包括程序运算引起的各种错误，如地址非法、检验错、页面失效、存取访问控制错、算术操作溢出、数据格式非法、除数为零、非法指令、用户程序执行特权指令、分时系统中的时间片以及用户态到核心态的切换等。

中断隐指令（硬件自动）

1.关中断 2. 保存断点 3.引出中断服务程序（实质就是取出中断服务程序入口地址传送给PC）

中断处理过程

1.关中断 2.保存断点 3.引出中断服务程序

4.保存现场和屏蔽字 5.开中断 6.执行中断服务程序 7.关中断 8. 恢复现场和屏蔽字 9.开中断，中断返回。中断服务程序的最后一条指令通常是一条中断返回指令，使其返回到原程序的断点处，以便继续执行程序。

1-3在cpu进行中断周期后，由中断隐指令（硬件自动）完成；4-9由中断服务程序完成

? DMA方式和中断方式的区别

① 中断方式是程序的切换，需要保护和恢复现场；而DMA方式除了预处理和后处理，其他时候不占用CPU任何资源。

② 对中断请求的响应只能发生在每条指令执行完毕；而对DMA请求的响应可以发生在每个机器周期结束时，只要CPU不占用总线时就可以响应

③ 中断传送过程需要CPU的干预；而DMA传送过程不需要CPU的干预，故数据传输速率非常高，适合于调整外设的成组数据传送。

④ DMA请求的优先级高于中断请求。

⑤ 中断方式具有对异常事件的处理能力，而DMA方式仅局限于传送数据块I/O操作。

⑥ 从数据传送来看，中断方式靠程序传送，DMA方式靠硬件传送。

? 什么是局部性原理?

序局部性原理，是指程序在执行时呈现出局部性规律，即在一段时间内，整个程序的执行仅限于程序中的某一部分。相应地，执行所访问的存储空间也局限于某个内存区域。

局部性原理又表现为：**时间局部性**和**空间局部性**。

时间局部性是指如果程序中的某条指令一旦执行，则不久之后该指令可能再次被执行；如果某数据被访问，则不久之后该数据可能再次被访问。

空间局部性是指一旦程序访问了某个存储单元，则不久之后。其附近的存储单元也将被访问。

复试面经

2018年3月12日 11:35

2018/3/12 11：36

一楼第一个 26 分钟，自我介绍，最喜欢的课，总结的一个都没问到，问了嵌入式，英语自我介绍，学得最好的课程，递归步骤，拓扑模型
四楼第一个 9min 自我介绍说了两句，老师说，口语不好，不让说了
第二个老哥 14min 口语还是自我介绍一分钟 抽了两道题 一个是什么烧绳子 一个是栈应用括号匹配
第三个 11min 自我介绍+家乡 主要问项目 自我介绍两句，介绍家乡，抽题，跨考老师不为难，聊项目
第四个直播：自我介绍省去了，直接介绍天津，抽题，还有介绍报考方向 可能也会问 家乡，自我介绍 本科网络，老师看了成绩单，问了了本科学的操作，网络，抽题是算法，里面老师四男一女，女的翻看材料 算法：有序数字找三个数和为 x
五号直播：没让自我介绍，介绍大学，抽题 c++ 老师出来拿水打断采访，说面试结束先撤
五个人 77 分钟。剩下的七个人时间要被压缩了。不用担心手写代码 因为没黑板 1030 开始 1130 结束七个人 每人平均八分半 开心
六号 个人陈述写的东西，好好想想 不要装逼 介绍大学，没了
Java 垃圾回收机制
2 个队列求公共子序列 中断机制和原因
无序数组 2 个相同 一个不同 找出来这个数
电通第一个 36min
工设 第一个 41min

2018/3/13 11：18

15 号 上午
一楼 YL 用英语问工作中用哪门需要做什么开发，完全没听懂
还有个问题，我做 golang 开发的，yl 老师让我答 golang 和 c++在做高并发是的区别，golang 为什么适合高并发 好像是特权指令内核态还是用户态
张三：我跨考，问我本科食品是学什么的 请说一下汉诺塔的模型 还有 PCB，问我英文是什么 数据库，事务是什么 英语，yl 让我介绍自己的学术背景 一楼多准备英语，yl 是让我用英文回答，本科学的什么
我就听到一个 academic 我也不知道 yl 再说啥 抽题，但老师主要让我说食品
抽题抽到的是 UNIX 系统的进程通信 然后一直在问实习做的项目十一点闲聊低级信号量。
然后存储区。这个分两类 共享数据结构 和 共享存储区 低级和高级 消息传递机制 有消息队列。描述一发。 管道 pipe 文件。unix 手法
四楼 一个括号匹配一个二叉排序书删除节点
四楼第二个，没让自我介绍，让我介绍学校，最多一分钟就被打断了
抽了链表求是否有环，问了烧绳子问题，看我数学低问了怎么求特征值，以及特征值有什么深层含义 进去老师都笑得可开心，张嘴说英文就眉头皱得特别厉害是跨考的，专业课没为难我
字神：自我介绍了一下为什么选云计算，两道题
一道两个单链表判断是否相交 一道饥饿 死锁 解释 其他什么也没说

智力问题

燃烧绳子问题.
烧一根不均匀的绳,从头烧到尾总共需要1小时.现在有若干条材质相同的绳子,问如何用烧绳的方法来计时一个小时十五分钟呢?

Solution1

需要三根绳子:
第一根和第二根同时点,第一根点一头,第二根点两头
在第二根燃完后（30分钟）,点燃第一根的另一头
前两根燃完后（15分钟）,点燃第三根的两头
全部燃完后（30分钟）
一共就是1小时15分钟.

Solution2

A,B,C三绳
A，C一头烧，B两头烧
B烧完时半小时，此时掐断C，A继续烧，A烧完时一小时，然后此时从两头烧C，烧完时时间为一小时十五分钟
同理，一小时45分钟也可以用类似的方式推出来~

3:15的时间时,时针与分针的夹角是多少度,怎么算的

3时15分=13/4时
3时15分时,分针与0点位置的夹角是(15/60)*360=90度
时针与0点位置的夹角是[(13/4)/12]*360=97.5度
所以在3时15分时,时针和分针的夹角
=97.5-90
=7.5度