# 程序设计

《我的第一本C+》 第1版第2次印刷

《C++程序设计(谭浩强)》 第1版第19次印刷

《C++ Primer第五版》 第1版第7次印刷

## 构成C语言的基本单位

函数

## 什么是程序

程序是为实现特定目标或解决特定问题而用计算机语言编写的指令序列

## 什么叫结构化程序设计

1）采用自顶向下、逐步求精及模块化的程序设计方法；

2）使用顺序、选择、循环三种基本控制结构构造

## 面向对象的三特征，五原则

面向对象特征：封装，继承，多态。分别解释（《第一本C++》 p349）

面向对象设计的SOLID原则

1) The Single Responsibility Principle 单一责任原则

当需要修改某个类的时候原因有且只有一个。

2) The Open Closed Principle 开放封闭原则

软件实体应该是可扩展，而不可修改的。

3) The Liskov Substitution Principle 里氏替换原则

当一个子类的实例应该能够替换任何其超类的实例时，它们之间才具有is-A关系。通俗地讲，就是子类型能够完全替换父类型，而不会让调用父类型的客户程序从行为上有任何改变

4) The Interface Segregation Principle 接口分离原则

不能强迫用户去依赖那些他们不使用的接口。换句话说，使用多个专门的接口比使用单一的总接口总要好。

5) The Dependency Inversion Principle 依赖倒置原则

1. 高层模块不应该依赖于低层模块，二者都应该依赖于抽象

2. 抽象不应该依赖于细节，细节应该依赖于抽象

## C与C++的不同

1）C语言是面向过程的，而C++是面向对象的，C++可以说是对c的扩充和增强

2）C和C++中都有结构的概念，但是在C语言中结构只有成员变量，而没成员方法，C++是在C的基础上加入了类的概念

此外，见 *在C++ 程序中调用被 C编译器编译后的函数，为什么要加 extern “C”*这个问题。

更多细节见《C专家编程》p266

## C++与C#的区别

C# 是一种完全面向对象的语言，而 C++ 不是

1. 继承

C#不允许多重继承（接口除外，接口可以多重继承），C++可以，因为C++有虚基类可以解决冲突问题

2. 抽象类

C++中包含抽象方法的类是抽象类，没有一个专用的关键字，C#中有abstract关键字，可用于修饰方法和类，抽象类不能被实例，这和C++一样

3. 关于接口

C++中没有接口，作为替代品，可以将所有的方法都定义为virtual XXX = 0,C#中有Interface

## C++和Java区别

1）指针。Java没有指针

2）多重继承。C++支持多重继承；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

3）自动内存管理。Java自动进行无用内存回收操作，不需要程序员进行删除

4）数据类型及类。Java是完全面向对象的语言，所有函数和变量部必须是类的一部分。除了基本数据类型之外，其余的都作为类对象；而C++允许将函数和变量定义为全局的

5）操作符重载。Java不支持操作符重载。

6）类型转换。Java不支持C++中的自动强制类型转换，如果需要，必须由程序显式进行强制类型转换。

7）Java不提供 goto语句，它虽然指定goto作为关键字，但不支持它的使用

8）C++能被操作系统直接运行，而Java需要JVM.

## C++与Java虚函数的区别

1. java中没有虚函数的概念，但是有抽象函数的概念，用abstract关键字表示
2. 在java中，所有方法默认就是虚拟的，只要方法不是声明为final类型的，那么肯定就是虚函数，不用为方法显示声明为virtual。
3. java里没有纯虚函数这个概念，只有interface。

详细的看：

http://zhidao.baidu.com/link?url=DLzRN17mwGshpMzaOfWEhn7GucEHBVaxPcwBwGwMk5lYisle\_ZBbOJDE2-r-tUd1F2l1ng752b-qaylNPrrSq41TB3J3k1vCe5C9omu9g3a

## 类和对象区别

类是对象的抽象，对象是类的具体实例

## malloc/free与new/delete的区别

一样在堆上分配内存

1. malloc与free是C++/C语言的标准库函数，new/delete是C++的运算符。它们都可用于申请动态内存和释放内存。
2. 对于非内部数据类型的对象而言，光用malloc/free无法满足动态对象的要求。malloc仅仅分配内存，free仅仅回收内存，并不执行构造和析构函数 。
3. malloc需要指定分配空间大小, 而new是自动计算的

new的空间大小限制：数组的总大小不得超过 0x7fffffff 字节

栈的大小通常在1M

对于简单数据类型（例如int [ ]），使用new分配内存后，可以使用free来释放内存。

调用析构函数可以清除给定的对象但是不会释放对象所在的空间。

free/delete对象后，对象空间的内容会清除变为0xeefe

## new失败后的处理

很久之前，C++编译器保留了C编译器的处理方式（malloc函数）：当operator new不能满足一个内存分配请求时，它返回一个NULL 指针。然而，随着技术的发展，标准的更新，编译器具有了更强大的功能，新时代的new在申请内存失败时：抛出一个bad\_alloc exception（异常）。

在Visual C++ 6.0 中目前operator new、operator new(std::nothrow) 和 STL不兼容、不匹配，而且不能完全被修复。如果在非MFC项目中使用Visual C++6.0中的STL，其即装即用的行为可能导致STL在内存不足的情况下让应用程序崩溃。

（http://book.51cto.com/art/201202/317767.htm）

## delete和delete[]

1. 带[]和不带的要配对使用，若不是，编译器可能不会给出警告，但其行为是未定义的。（《C++ peimer》 p425）
2. int a[100]; delete a; 会内存泄漏吗?不会。

class A {public: int a[100]; }; A\* pa = new A[100]; delete pa; 呢？也不会。

方括号的存在会使编译器获取数组大小（size）然后析构函数再被依次应用在每个元素上，一共size次。否则，只有一个元素被析构。

1. 无论哪种情况，分配的全部空间被返还给自由存储区。delete()和delete[]()都是传入一个void \*参数，不是根据参数类型来释放空间的。（《C++ primer》 p727）
2. 以上可见，释放数组空间和为数组调用析构函数是两个独立的部分。即使不写delete，也会将所有空间释放，其机制可以是前置的长度信息，也可以不是（如后置的特征分割符等）。

## new和delete重载

使用new分配某个重载了new的类的对象空间时，先调用重载的new，再调用构造函数。

使用了delete释放某个重载了delete的类的对象空间时，先调用析构函数，然后再调用重载的delete。

（http://blog.163.com/hbu\_lijian/blog/static/126129153201211510361484/）

更多内容见《C++ Primer》p726

## const int \* p ,int const \* p ,int \* const p区别

（具体略）第三个声明时必须赋初值

## const和#define相比，有何优点

1）const常量有数据类型，而宏常量没有数据类型。编译器可以对前者进行类型安全检查。而对后者只进行字符替换，没有类型安全检查，并且在字符替换可能会产生意料不到的错误（边际效应）。

2）有些集成化的调试工具可以对const常量进行调试，但是不能对宏常量进行调试。

## typedef和define有什么区别

（1）用法不同：typedef用来定义一种数据类型的别名，增强程序的可读性。define主要用来定义常量，以及书写复杂使用频繁的宏。

（2）执行时间不同：typedef是编译过程的一部分，有类型检查的功能。define是宏定义，是预编译的部分，其发生在编译之前，只是简单的进行字符串的替换，不进行类型的检查。

（3）作用域不同：typedef有作用域限定。define不受作用域约束，只要是在define声明后的引用都是正确的（若写在{}中，在“}”之后也是有效的 ）。

（4）对指针的操作不同：typedef和define定义的指针时有很大的区别。

typedef char \*pStr1; pStr1 s1, s2;

#define pStr2 char \* pStr2 s3, s4; //s4为char型

*注意*：typedef定义是语句，因为句尾要加上分号。而define不是语句，千万不能在句尾加分号。

## 关键字struct和class区别

结构体和类都是提供了数据的封装和构造

1）默认的访问控制

2）struct的默认继承方式为public，而class的默认继承为private（struct和class间可以相互继承）

3）“class”这个关键字还用于定义模板参数，就像“typename”。但关键字“struct”不用于定义模板参数。

## 引用和指针区别

相同点：

都是地址的概念

不同点：

A. 指针是一个实体，而引用仅是个别名；

B. 引用只能在定义时被初始化一次，之后不可变；指针可变

C. 引用不能为空，指针可以为空；

D. “sizeof引用”得到的是所指向的变量(对象)的大小，而“sizeof 指针”得到的是指针本身的大小；

E. 指针和引用的自增(++)运算意义不一样；

F. 引用是类型安全的，而指针不是 (引用比指针多了类型检查)

程序在编译时分别将指针和引用添加到符号表上，符号表上记录的是变量名及变量所对应地址。指针变量在符号表上对应的地址值为指针变量的地址值，而引用在符号表上对应的地址值为引用对象的地址值。符号表生成后就不会再改，因此指针可以改变指向的对象（指针变量中的值可以改），而引用对象不能改。

## sizeof和strlen的区别

1. sizeof是一个操作符，strlen是库函数。
2. sizeof的参数可以是数据的类型，也可以是变量，而strlen只能以结尾为‘\0‘的字符串作参数。
3. 编译器在编译时就计算出了sizeof的结果。而strlen函数必须在运行时才能计算出来。并且sizeof计算的是数据类型占内存的大小，而strlen计算的是字符串实际的长度。
4. 数组做sizeof的参数不退化，传递给strlen就退化为指针了。

## 在C++ 程序中调用被 C编译器编译后的函数，为什么要加 extern “C”

答：C++语言支持函数重载，C语言不支持函数重载。函数被C++编译后在库中的名字与C语言的不同。

假设某个函数的原型为： void foo(int x, int y); 该函数被C编译器编译后在库中的名字为\_foo，而C++编译器则会产生像\_foo\_int\_int之类的名字。 C++提供了C连接交换指定符号extern“C”来解决名字匹配问题。

## 什么是内存泄露，内存溢出，段错误，堆栈溢出

内存泄露（memory leak）：程序动态分配了内存，但是在使用完毕后没有释放这部分内存，从而导致一直占据该内存单元的情况

内存溢出（out of memory）：所需要的内存远远超出了主机内安装的内存所承受大小

段错误（segmentation fault）：访问的内存超出了系统所给这个程序的内存空间

堆栈溢出（stack overflow）向缓冲区中写入超过其本身长度的数据，溢出而使得造成缓冲区以外的存储单元被改写

## 什么是多态

简单点说:“一个接口，多种实现”

多态性在C++中都是通过虚函数（Virtual Function）实现的

## 什么是MVC，为什么要用MVC

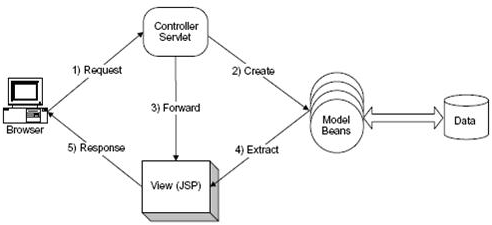
模型(model)－视图(view)－控制器(controller)的缩写，是一个框架模式。

使应用程序的输入、处理和输出分开。

用一种业务逻辑和数据显示分离的方法组织代码，将业务逻辑被聚集到一个部件里面，在界面和用户围绕数据的交互能被改进和个性化定制的同时而不需要重新编写业务逻辑。

框架和设计模式区别：框架通常是代码重用，而设计模式是设计重用。

## MVC框架



Web开发框架：

structs2主要用于view，前台框架

spring主要用于control，后台框架

hibernate主要用于model，持久层框架

## 进程内存布局（一个C、C++程序编译时内存分为5大存储区）

堆区(heap)、栈区(stack)、全局/静态区、文字常量区、程序代码区(code segment)。

[ 代码区 ]

[ 静态数据区 ]

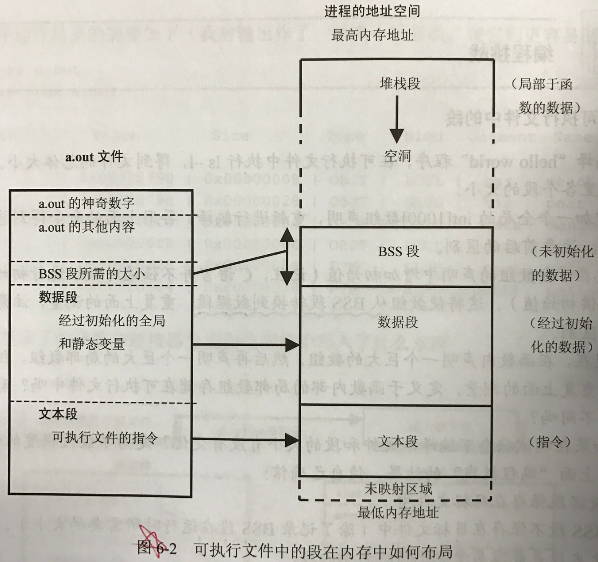
[ 栈 ] ↓ 高地址

[ ]

[ 堆 ] ↑ 低地址

（VS10中有时这样，有时不这样）

linux可执行文件内存布局：（《C专家编程》p120）



## include包含文件查找的顺序

#include “”

1. 当前文件所在的文件夹内
2. 编译器设置的include路径
3. 系统的INCLUDE环境变量

#include <> 为后两步。

## volatile

（《C++ primer》p757）

当对象的值可能在程序的控制或检测之外被改变时，应该将该对象声明为volidate。volidate告诉编译器不该对这样的对象进行优化。

（http://www.cnblogs.com/yc\_sunniwell/archive/2010/07/14/1777432.html）

当要求使用 volatile 声明的变量的值的时候，系统总是重新从它所在的内存读取数据，即使它前面的指令刚刚从该处读取过数据。

## inline

（题目：http://www.nowcoder.com/test/question/done?tid=4814418&qid=44774）

1. 内联函数应该在头文件中定义
2. 内联函数定义也可以放在源文件中，但此时只有定义的那个源文件可以用它！
3. 类内定义的成员函数自动的成为内联函数。（《C++ 谭浩强》p245）
4. 非类内定义的成员函数，不会默认为内联函数
5. 若在类体外定义了inline成员函数，必须将类定义和成员函数的定义放在同一个头文件中。（《C++ 谭浩强》p246）

## extern，register，explicit，final，auto，decltype

extern

1. 声明全局变量/函数。提示编译器遇到此变量和函数时在其他地方中寻找其定义。分同一文件（提前引用声明）和不同文件（外部变量声明）（《C++程序设计(谭浩强)》P119）。
2. 用于函数定义时，表示全局可见，定义时省略extern默认为外部函数。（p125）
3. extern语句若包含初始值就不再是声明，而变成定义：extern int a=1;（《C++ Primer》p46）
4. 在函数体内，不能初始化extern标记的变量（《C++ Primer》p46）

说白了，定义全局变量/函数默认为extern，extern不能用于局部变量定义。（？）

register声明寄存器变量，对编译系统只是建议性的，其实没有必要。

explicit：只对构造函数起作用，用来抑制隐式转换。（《C++ Primer》p265）

final：C++11中防止被继承

auto：与const的关系见《C++ Primer》p62

decltype：在const上与auto的不同

## static

1. 静态局部变量。函数调用结束后不消失而保留原值。（《C++程序设计(谭浩强)》P117）

*注意*：static定义的变量分配在静态区，所以其定义的变量的默认值为0，普通变量的默认值为随机数

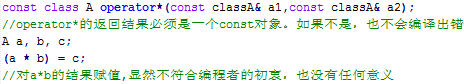
1. 静态外部变量，只能用于本文件（《C++程序设计(谭浩强)》P121）
2. 内部函数/静态函数，使函数只局限于所在文件。如果在不同的文件中有同名的内部函数，互不干扰。（《C++程序设计(谭浩强)》P125）
3. 在类中的static成员变量意味着它为该类的所有实例所共享
4. 在类中的static成员函数属于整个类所拥有，这个函数没有this指针，因而只能访问类的static成员变量。

PS：在java中还有静态代码块

## const

意味着不能修改。

1. 对指针来说，可以指定指针本身为const，也可以指定指针所指的数据为const。
2. 在一个函数声明中，const可以修饰形参。
3. 对于类的成员函数，若指定其为const类型，则表明其是一个常函数，不能修改类的成员变量
4. 对于类的成员函数，有时候必须指定其返回值为const类型，以使得其返回值不为“左值”。如下图：



对已经定义了的const对象再次赋值会编译错误。

## 非静态成员函数是否可以直接访问静态成员函数

非静态可以直接访问静态；静态不可以直接访问非静态。

因为静态函数可以直接通过“类::函数”方式调用，不一定要通过对象来调用函数；而非静态函数必须通过对象来调用，不能用“类::函数”方式。

## 为什么static成员函数不能声明为const

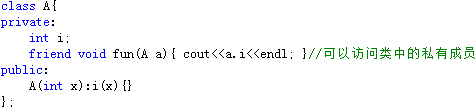
静态成员函数类似于静态数据成员，没有this指针，不能访问本类中非静态成员。（《谭浩强C++》 p298）

用const修饰类的成员函数，写在函数的最后表示不能修改成员变量，不是指写在前面表示返回值为常量。为了确保该函数不能修改类的实例的状态，会在函数中添加一个隐式的参数const this\*，也就是说该函数必须含有this指针。

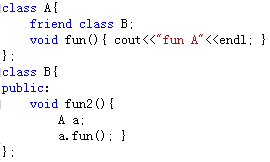
## 友元

1. 友元函数：可以访问另一个类的所有成员（不论私有、共有、变量、函数）。

要访问非static成员时，需要对象做参数，如下图（不使用参数，在函数内创建对象也是可以的）；要访问static成员或全局变量时，则不需要对象做参数。



1. 友元类：可以访问另一个类的所有成员。



友元的关系是单向的，不能传递。

缺点：友元函数破环了封装机制。

## 友元函数，友元类

在友元函数或友元类中，可以直接访问类的protected或private成员。

但一个类以protected方式继承另一个类，不能访问private成员。

## 匿名命名空间

将作用域限定在本文件范围。全局静态声明能实现同样效果。

## 内部类

一种在类声明里面定义的一种局部数据类型。

内部类的声明有public和private之分：

如果声明为public，那么外面也可以用它来定义变量，比如Outer::Inner var。

如果声明为private，那么外面不能用来定义变量。

class Outer{

int m\_o;

public:

class Inner{

int m\_j;// Outer对Inner中对象的访问权限不会因为是内部类而不同

public:

int m\_i;

Inner(){}

Inner(Outer m\_outer,int x){ m\_outer.m\_o=x; }//访问Outer对象

};

Outer(int y){ m\_inner.m\_i=y;}

void Display(){ cout<<m\_o<<endl<<m\_inner.m\_i<<endl; }

Inner m\_inner;

};

## 什么是预编译

正式编译之前，处理代码中以“#”开头的指令，如头文件拷贝、宏定义替换、条件编译等。

## 全局对象、静态对象、局部栈对象、局部堆对象的构造和析构

全局对象的构造函数在main函数之前调用，析构函数在main函数之后调用。

局部静态对象在定义的时候调用构造，main函数之后调用析构，但比全局对象的析构函数早。

局部栈对象在定义的时候调用构造函数，出了可见范围的时候调用析构函数。

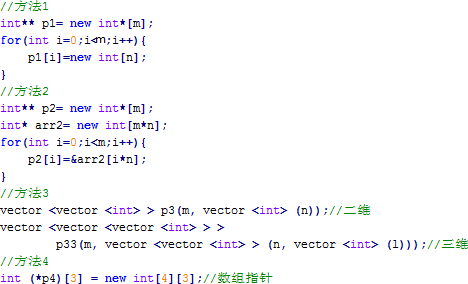
堆对象在new的时候调用构造函数，delete的时候调用析构，不delete空间不会自动释放。

## 模板的类型参数和非类型参数，非类型参数使用注意事项

非类型参数表示一个值，必须是常量表达式，从而允许编译器在编译时实例化

## 动态申请二维数组

以下方法都可以使用p[i][j]访问（已测试过）。



方法2相对于方法1分配内存更快。

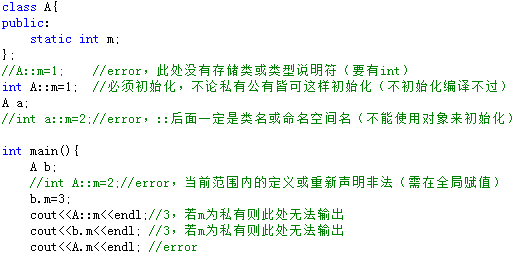
方法3中所有值为0。

方法4中数字3不能为变量！

指针数组对应的是方法一中int \*[m]；数组指针对应方法4。

## 成员变量的初始化

1. 成员变量在使用初始化列表初始化时，与初始化成员列表的顺序无关，只与定义成员变量的顺序有关（先定义的先初始化）。
2. 类成员在定义时，是不能初始化的（C++11已支持定义时初始化）；类中const成员常量必须在构造函数初始化列表中初始化；类中static成员变量，必须在类外初始化，不管是不是private的，并且如下图所示要有类型说明。



1. 整形类别的静态常量数据成员可以在类内初始化(即类内声明的同时初始化)，也可以在类外。非整型则不行！

class A {

public:

static const int m ;

};

const int A::m = 1;

## 类的构造和析构顺序

构造顺序：

基类成员变量构造 --> 基类构造函数 --> 子类成员变量构造 --> 子类构造函数

析构顺序：完全反的！

子类析构函数 --> 子类成员变量析构 --> 基类析构函数 --> 基类成员变量析构

## 多继承的构造顺序

(http://www.cnblogs.com/cppfans/articles/1713543.html)

class Derived :public Base1, virtual public Base2, public Base3,

virtual public Base4{

public:

Derived() :Base4(), Base3(), Base2(), Base1(), obj2(), obj1() { }

OBJ1 obj1; OBJ2 obj2;

};

顺序为：Base2，Base4，Base1，Base3，OBJ1，OBJ2

析构顺序完全相反。

按下列顺序被调用：

(1)虚基类按照被继承的顺序构造（即声明顺序）

(2)非虚基类按照被继承的顺序构造；

(3)成员对象按照声明的顺序调用；

(4)类自己的构造函数。

## 派生类构造函数

如果使用初始化列表且对基类赋值，一般形式为：

派生类构造函数名(总参数列表): 基类构造函数名(参数列表)

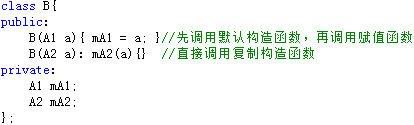
{ 派生类中新增数据成员初始化语句 }

也可以用《谭浩强C++》p366中，把{ }中赋值全放到初始化表中

基类如果只有有参的构造函数，则必须使用初始化列表。

## 类的构造函数：拷贝和赋值

1. 以下代码中，方法2的效率更高（不去考虑B()中传参那次构造）。



在一个class实例化之前，所有的member都会被初始化，如果member是个class，那么那个class的constructor就会被调用。上面代码使用“=”，总共会调用2次A1的构造。

比较标准的方式是使用初始化列表：B(const A1 &a):mA1(a){}

上面说明看不明白的话，看下面例子：

class A{

public:

A() { cout<<"A"<<endl; }

A(int i) { cout<<"A:"<<i<<endl; }

A(const A &a){ cout<<"Ac"<<endl; }

};

class B{

A x;

public:

B(){ cout<<"B"<<endl; }

B(int i):x(i){ cout<<"B:"<<i<<endl; }

B(const A &a):x(a){ cout<<"Ba"<<endl; }

};

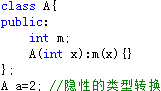
A a;

B b; //A B ,成员x被初始化

B b1(2); //A:2 B:2

B b2(a); //Ac Ba

1. 以下代码，把数值2转换成class A，因此会调用A(2)，然后用operator =符值给a。



若不想这样，可以explicit A(int x){}，禁止隐性类型转换。

1. 函数返回值

class A{

public:

int x;

A(int x){ this->x=x; cout<<"construct:"<<x<<endl; }

A(A &a){ x=a.x+1; cout<<"copy"<<endl; }

~A(){ cout<<"destruct:"<<x<<endl; }

A operator =(A a){ x=a.x+10; cout<<"assign"<<endl; }

};

A fun(A a){ return a; }

A a1(1);

A a2=fun(a1);

//construct:1

//copy

//copy

//destruct:2

//destruct:3

//destruct:1

fun()返回结果时先对a2进行拷贝构造（而不是赋值，虽然是等号），之后再析构返回值。如果是先析构返回值，就没法构造a2了。

## 虚函数，纯虚函数，抽象类

**虚函数**：被virtual关键字修饰的成员函数。

虚函数作用：实现多态性（Polymorphism），多态性是将接口与实现进行分离

基类指针调用虚函数，将调用这个指针所指向的具体对象的虚函数。

父类A有virtual关键字，子类B没有virtual关键字，子类的子类C也没有virtual关键字，A \* c=new C(); c调用虚函数指向的是c的函数。也就是说vitual关键字是可选的，可能需要追溯到继承体系的源头才能确定某个方法是否是虚函数。

**纯虚函数**：virtual void fun()=0; 基类中没有定义，但要求在它的派生类中定义自己的版本

**抽象类**：至于少有一个纯虚函数的类，抽象类提供了处理各种不同派生类的统一接口，将实现的责任交给了派生类。

不能创建抽象类的具体对象，必须实现其中的纯虚函数才能成为一个实体类。

## 虚继承，虚基类

默认情况下，如果某个类在派生过程中出现了多次，则派生类中将包含该类的多个子对象。我们通过**虚继承**（virtual inheritance）机制解决这个问题。共享的基类子对象称为**虚基类**（virtual base class）。

具体代码怎样写见《C++ primer》p717：

class A{};

class B: public virtual A{};

class C: virtual public A{};

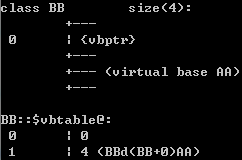
class D: public B, public C{};

对于虚继承的类的空间：

class AA{};

class BB:public virtual AA{};

cout<<sizeof(BB); //4



也是要用到虚表和虚指针的。

## 简述多态实现的原理

编译器发现一个类中有虚函数，便会立即为此类生成虚函数表 vtable。虚函数表的各表项为指向对应虚函数的指针。编译器还会在此类中隐含插入一个指针vptr（对vc编译器来说，它插在类的第一个位置上）指向虚函数表。调用此类的构造函数时，在类的构造函数中，编译器会隐含执行vptr与vtable的关联代码，将vptr指向对应的vtable，将类与此类的vtable联系了起来。另外在调用类的构造函数时，指向基础类的指针此时已经变成指向具体的类的this指针，这样依靠此this指针即可得到正确的vtable，。如此才能真正与函数体进行连接，这就是动态联编，实现多态的基本原理。

## 构造与析构函数可以是虚函数吗

构造函数不能是虚函数。vs10中编译不通过。

A，虚函数的作用在于通过父类的指针或者引用来调用它的时候能够变成调用子类的那个成员函数。而构造函数是在创建对象时自动调用的，不可能通过父类的指针或者引用去调用。

B，虚函数对应一个vtable，存储在对象的内存空间，在构造函数调用后才建立？，如果构造函数是虚函数，那必然需要通过vtable来找到虚构造函数的入口地址，申请的内存还没有做任何初始化，不可能有vtable的。

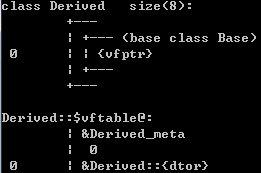
C，防止子类中重写跟父类构造函数一样的函数,将父类的构造函数覆盖。

//////

析构函数在某些情况下必须为虚函数（比如你想让你类能够被继承，那么这个类的析构函数最好是虚的）。

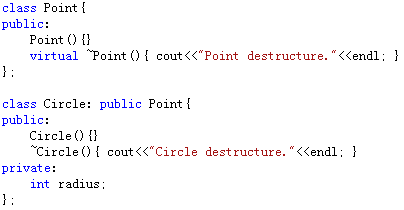
释放指针的合理过程是: :先释放继承类的资源,再释放基类资源。

基类的析构函数如果不是虚函数，基类指针无法操作继承类的成员，我们往往通过基类的指针来销毁对象，只是释放了基类的资源，而没有调用继承类的析构函数。（若不是虚函数，则派生类对象空间中不会有派生类析构函数的地址，当用基类指针释放对象时，在派生类对象空间中不存在指定的析构函数，只能根据声明到基类对象共用的基类成员函数中找析构函数）（析构函数在类对象空间中是{dtor}，而不是类的名字。dtor：destructor）



（上图size为8，但sizeof(Derived)却为4，why？）

当类里面有定义虚函数的时候,编译器会给类添加一个虚函数表,里面来存放虚函数指针,这样就会增加类的存储空间.所以,只有当一个类被用来作为基类的时候,才把析构函数写成虚函数。（注意：virtual是写在基类的析构函数前，而不是派生类函数前，虽然这两个函数不同名！）



对于下面这种，必须要有delete才会析构！

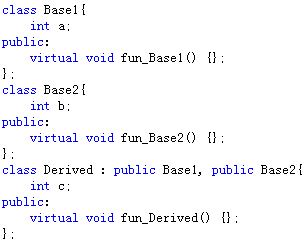
Point \*p=new Circle;

delete p;

对于下面这种，作用域结束后自动析构。

Circle c;

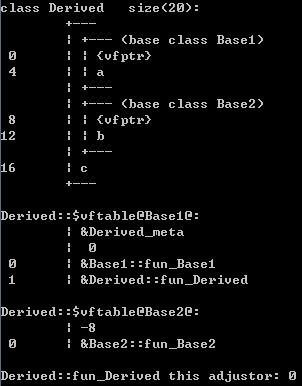
## 一个类继承自其它两个类时的内存布局



如下图，可以看出Derived类中先后为Base1，Base2，Derived自身成员变量。

fun\_Derived在vftable@Base1中。（第一个继承过来的虚函数表下面）

如果继承关系为菱形结构，使用虚继承，会有{vbptr}，表示虚继承的虚基类指针。



从上图可以看出，类对象的空间大小为20，分别为3个变量和2个vfptr。虚函数在vftable中，不会占用对象的空间。若类有其他成员函数，是所有对象共用的，也不会出现在对象空间中。

## sizeof空间

1. 为了方便对结构体内元素的访问和管理，当结构体内元素的长度都小于处理器的位数时，便以结构体中最长的数据元素为对齐单位；否则，以处理器位数为对齐单位，如VC6.0默认为8。
2. 在VC中，可以用#pragma pack(?)来设置对齐调整。
3. 空类（结构体）大小为1，每个实例在内存中都要有一个独一无二的地址。
4. 可以用函数做参数

short f();

cout<<sizeof(f()); //结果为sizeof(short)，2

1. sizeof后如果是类型必须加括号，如果是变量名可不加。因为它是操作符而不是函数。

## C++的空类有哪些成员函数

缺省构造函数。

缺省*拷贝构造函数*。X::X(const X& x)

缺省析构函数。

缺省*赋值运算符*。

缺省取址运算符。

缺省取址运算符 const。

注意：有些书上只是简单的介绍了前四个函数（如Effective C++ 第三版p34）。没有提及后面这两个函数。但后面这两个函数也是空类的默认函数。

## 谈谈你对拷贝构造函数和赋值运算符的认识

拷贝构造函数和赋值运算符重载有以下两个不同之处：

（1）拷贝构造函数生成新的类对象，而赋值运算符不能。

（2）由于拷贝构造函数是直接构造一个新的类对象，所以在初始化这个对象之前不用检验源对象是否和新建对象相同。而赋值运算符则需要这个操作，另外赋值运算中如果原来的对象中有内存分配要先把内存释放掉（默认没有释放new的内存；只是将类的成员赋值了，并没有复制new的内存）

注意：当有类中有指针类型的成员变量时，一定要重写拷贝构造函数和赋值运算符，不要使用默认的。

## 析构函数能否抛出异常

C++并没有禁止析构函数引发异常，但是C++不推荐这一做法。

1. 如果析构函数抛出异常，则异常点之后的程序不会执行，如果析构函数在异常点之后执行了某些必要的动作比如释放某些资源，则这些动作不会执行，会造成诸如资源泄漏的问题。

2. 通常异常发生时，c++的机制会调用已经构造对象的析构函数来释放资源，此时若析构函数抛出异常后若没有被正确捕获，系统将会导致调用标准库terminate函数。通常terminate函数将调用abort函数，导致程序的非正常退出。

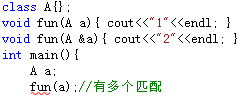
（《C++ Primer》p685）

## 重写（override）（~~覆盖~~），重载（overload）和隐藏的区别

1. 重写是指子类重定义父类虚函数的方法。
2. 重载是指允许多个重名函数，但参数列表不同。

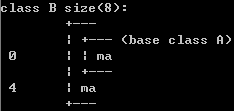
重载是静态的，在编译期间就绑定了；重写的函数调用在编译期是无法确定的，在运行期才绑定。

引用和非引用两种情况不算重载！



1. 隐藏是指派生类的函数屏蔽了与其同名的基类函数。注意只要同名函数，不管参数列表是否相同，基类函数都会被隐藏。（基类成员变量也会有隐藏这种情况）

虽然被隐藏，但内存中是存在的，可以这样 b.A::ma 访问到。



http://www.jb51.net/article/54225.htm

## “引用”与多态的关系？

引用是除指针外另一个可以产生多态效果的手段。这意味着，一个基类的引用可以指向它的派生类实例。

例class A; class B : class A{…}; B b; A& ref = b;

# 程序设计-高级

## 右值引用与与转移语义

右值引用（rvalue reference）：减少不必要的临时对象的创建、拷贝以及销毁。通过&&获得右值引用。

move函数：获得绑定到左值上的右值引用



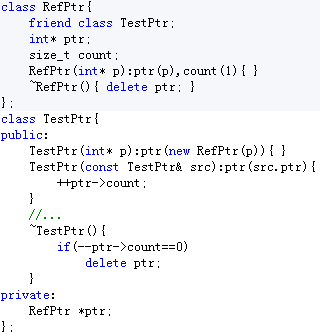
## 智能指针(smart pointer)

存储指向动态分配（堆）对象指针的类，用于生存期控制，能够确保自动正确的销毁动态分配的对象，防止内存泄露。

（http://blog.csdn.net/hackbuteer1/article/details/7561235）

管理指针成员三种方法1）复制时只复制指针，不复制指针指向的对象。当其中一个指针把其指向的对象的空间释放后，其它指针都成了悬浮指针。2）复制指针指向的对象。这样可能造成空间的浪费。3）折中的方式。利用一个辅助类来管理指针的复制。原来的类中有一个私有成员指针指向辅助类，辅助类的数据成员是一个计数器和一个指针，把原来类声明为友元类。

（int \* ptr 是智能指针指向的对象）

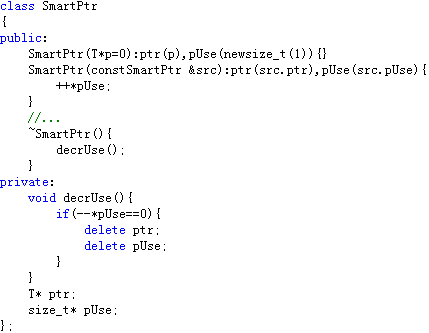


上面第三种是智能指针的一种实现方式，另一种方法：

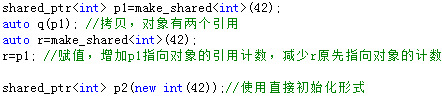
（http://baike.baidu.com/link?url=PszrjGcmRQBBDC4df18HZLSSzTQfCVpw-plXOmFnka1BTvnpRdbbeA6SzOjJu-dLPVScvzcUwkb9RkQvfASvJq）

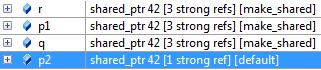
避免上面方案中每个使用指针的类自己去控制引用计数，可以用一个类把指针封装起来。我们把这样的类叫句柄类（句柄类是存储指向动态分配（堆）对象指针的类）。在封装句柄类时，需要申请一个动态分配的引用计数空间。（注意，不要和单例模式搞混）

（T \* ptr 是智能指针指向的对象）



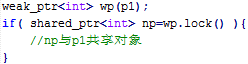
1. shared\_ptr 允许多个指针指向同一个对象





r赋值p1前引用为1。

1. unique\_ptr 独占指向的对象。某个时刻只能有一个unique\_ptr指向给定对象，当unique\_ptr被销毁时，指向对象也被销毁。
2. 它没有make\_shared函数，只能采用直接初始化形式
3. 不支持拷贝和赋值，但可以通过release或reset转移所有权
4. weak\_ptr 是一种弱引用，指向shared\_ptr所管理的对象
5. 不会改变shared\_ptr的引用计数和对象释放
6. 不能使用weak\_ptr直接访问对象，而必须使用lock



## 显示类型转换

《C++ Primer第五版 p144》

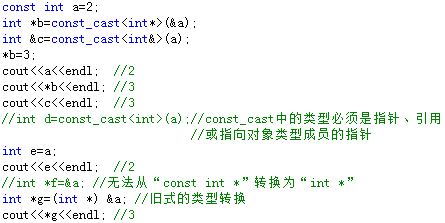
旧式类型转换：

type (expr) //函数形式

(type) expr //C语言风格

当执行旧式转换合法时，类似static\_cast和const\_cast；不合法时，类似reinterpret\_cast。

1. static\_cast：具有明确定义的类型转换，只要不包含底层const
2. const\_cast：将常量对象转为非常量对象（去掉const性质）



1. reinterpret\_cast：通常为运算对象的位模式提供较低层次上的重新解释



1. dynamic\_cast 《C++ Primer第五版 p730》

用于基类和派生类的指针/引用间转换

原理：RTTI（run-time type identification，运行时类型识别）

除了dynamic\_cast以外的转换，都是在编译期就得以确定的。

## Vector空间分配

1）创建空vector对象时，容量为0。

2）当现有的容量用完后，编译器才会重新分配空间。

3）VC6.0平台下，新存储空间的增量为旧存储空间容量的1倍。

VS2010平台下，新存储空间的增量为旧存储空间容量的1/2，即capacity的增长顺序为：1、2、3、4、6、9、13、19、28、42、63

4）当需求大小小于当前容量时，容器不会退回内存空间。（《C++ primer》p318）

1、调用reserve，capacity将会>=传递给reserve的参数

2、resize改变的是元素数目，而非容器容量

3、调用shrink\_to\_fit请求将capacity减小为size大小，标准库不保证退还内存

5）在VS10中：vector在栈上的部分有一个指针指向分配在堆上的空间。对于string，分配大小不大于15时，直接在栈上，大于15后，同vector。

## STL内存管理

http://www.tuicool.com/articles/FFvY7nR

为了解决小型区块所可能造成的内存破碎问题，SGI（stl比较有名的三个实现之一）设计了双层级配置器：当配置区块超过128 bytes 时，视为足够大，调用第一级配置器，直接使用malloc()和free()配置和释放内存空间；当小于128 bytes 时，调用第二层级配置器，采用memory pool的整理方式。

memory pool管理：每次配置一大块内存的时候，都对应维护一个自由链表（free-list）。下次若有相同大小的内存需求时，直接从free-list中取，当有内存被释放时，回收到free-list当中。第二级配置器维护16个free-list，分别管理8,16,24...128 bytes 大小的区块，客户端请求的区块会自动调整到上述大小，如 20 bytes 调整到 24 bytes。

（详细说明：http://www.cnblogs.com/dwdxdy/archive/2012/06/05/2537208.html）

使用内存池的优点有：降低动态申请内存的次数，提升系统性能；减少内存碎片，增加内存空间使用率。

## 如何实现多个线程同步

Java：1）synchronized关键字 2）Metux互斥体

CPP：（暂缺）

## linux C++中fork（）函数

一个进程调用fork（）函数后，系统先给新的进程分配资源，例如存储数据和代码的空间。然后把原来的进程的所有值都复制到新的新进程中，只有少数值与原来的进程的值不同。相当于克隆了一个自己。

## 内核能调用标准库函数吗

不能。

当编写驱动程序时，一般情况下不能使用C标准库的函数。linux内核也提供了与标准库函数功能相同的一些函数，但二者还是稍有差别。

## 静态链接库和动态链接库比较

（《VC++深入详解》p701）

**静态库**：函数和数据被编译进一个二进制文件（通常为.lib）。编译连接可执行文件时，连接器从库中复制这些函数和数据并和应用程序的其他模块组合起来创建最终的EXE文件。

**动态库**：一个引入库lib文件和DLL文件，前者包含导出的函数和变量的符号名，后者包含实际的函数和数据。编译可执行文件时只需引入库文件，运行时才加载DLL文件。

静态链接库的优点：

(1) 代码装载速度快，执行速度略比动态链接库快；

(2) 只需保证在开发者的计算机中有正确的.LIB文件，在以二进制形式发布程序时不需考虑在用户的计算机上.LIB文件是否存在及版本问题，可避免DLL地狱等问题。

动态链接库的优点：

(1) 更加节省内存并减少页面交换；

(2) DLL文件与EXE文件独立，只要输出接口不变（即名称、参数、返回值类型和调用约定不变），更换DLL文件不会对EXE文件造成任何影响，因而极大地提高了可维护性和可扩展性；

(3) 不同编程语言编写的程序只要按照函数调用约定就可以调用同一个DLL函数；

(4) 适用于大规模的软件开发，使开发过程独立、耦合度小，便于不同开发者和开发组织之间进行开发和测试。

不足之处：

(1) 使用静态链接生成的可执行文件体积较大，包含相同的公共代码，造成浪费；

(2) 使用动态链接库的应用程序不是自完备的，它依赖的DLL模块也要存在，如果使用载入时动态链接，程序启动时发现DLL不存在，系统将终止程序并给出错误信息。而使用运行时动态链接，系统不会终止，但由于DLL中的导出函数不可用，程序会加载失败；速度比静态链接慢。

## MFC消息机制

（腾讯校招面试）

http://blog.csdn.net/ouyang\_linux007/article/details/7638434

# Java

## Java中两种实现多线程的方式以及两种方式之间的区别

一是直接继承Thread类，二是实现Runnable接口

PS：三是使用ExecutorService、Callable、Future实现有返回结果的多线程

## 同步机制

synchronized，lock

## 接口和抽象类的区别

一个类只能继承一个类（抽象类）(正如人不可能同时是生物和非生物)，但是可以实现多个接口(吃饭接口、走路接口)。

1. 抽象类和接口都不能直接实例化，如果要实例化，抽象类变量必须指向实现所有抽象方法的子类对象，接口变量必须指向实现所有接口方法的类对象。
2. 抽象类要被子类继承，接口要被类实现。
3. 接口只能做方法申明，抽象类中可以做方法申明，也可以做方法实现。
4. 接口里定义的变量只能是公共的静态的常量，抽象类中的变量是普通变量。
5. 抽象类里的抽象方法必须全部被子类所实现，如果子类不能全部实现父类抽象方法，那么该子类只能是抽象类。同样，一个实现接口的时候，如不能全部实现接口方法，那么该类也只能为抽象类。
6. 抽象方法只能申明，不能实现。abstract void abc();不能写成abstract void abc(){}。
7. 抽象类里可以没有抽象方法
8. 如果一个类里有抽象方法，那么这个类只能是抽象类
9. 抽象方法要被实现，所以不能是静态的，也不能是私有的。
10. 接口可继承接口，并可多继承接口，但类只能单根继承。

## 线程安全的集合对象

ArrayList线程不安全，Vector线程安全；

HashMap线程不安全，HashTable线程安全；

StringBuilder线程不安全，StringBuffer线程安全。

Stack 继承自Vector 也是线程安全的。

（记忆：箭头在桌上被缓冲了，安全）

## HashMap初始容量、加载因子

初始容量为16，加载因子为0.75

# 数据结构

复习以南师教材为主，王道书（2014年）为辅。

## 使用类模板的链表的Node结构和类的结构

略

## KMP算法(串的模式匹配)

可以练习Leetcode28 Implement strStr()

## 复习排序代码，尤其是快排

略

## 复习树，图代码

数：

中序的非递归实现 （《王道》p113）

二叉树的构造：单个序列，两个序列，拷贝 （《教材》p100）

二叉树的查找 和 二叉排序树的查找，注意两者区别（《教材》p106&179）

二叉排序树的插入（《教材》p178）

图：

最短路径，最小生成树

## 线性表

1. 线性表：数据元素之间仅具有单一前驱和后继关系。双向链表和循环链表也是线性表，但关联数组不是。
2. 线性表的链式表示引入头结点的优点：1）第一个位置上的操作和在表的其他位置上的插入删除操作一致，无须进行特殊操作。2）无论链表是否为空，头指针都是指向头结点的非空指针。
3. 栈和队列是操作受限的线性表。
4. 栈的链式存储：新增节点指向旧节点，不需要使用头结点
5. 队列的链式存储：新结点插入链表的尾部，旧结点指向新结点
6. 队列的顺序存储：初始时front和rear指向哪个？队列满的判断？（《教材》p49）

## 时间复杂度推导

T(n)=O(1) if n=1;

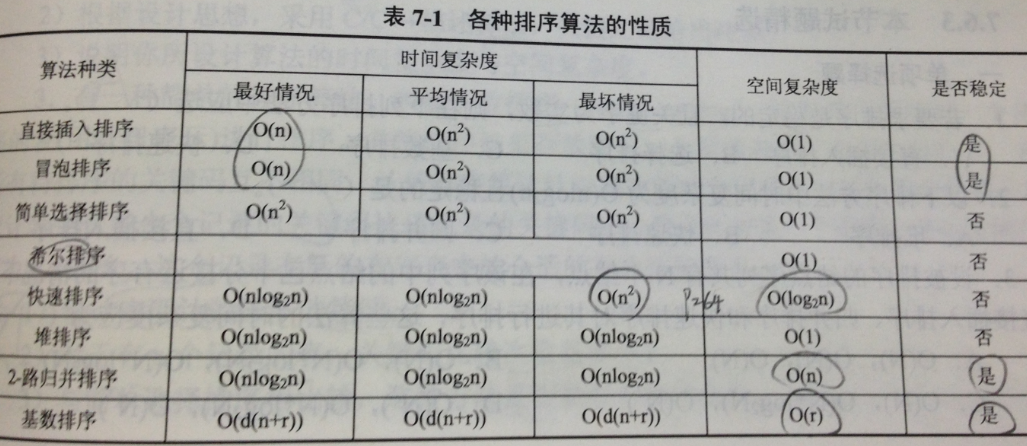
T(n)=2\*T(n/2)+O(n) if n>1;

so T(n) is O(nlgn).

推导过程见《王道》p8，令n=2^k。或者直接用n来推导。

时间复杂度要去掉系数，写为log(n)。

## 排序时间复杂度



（助记：插入泡泡，做二选题，心态不稳，题目是如何快速的将一堆数据进行归并，快速的堆是不稳定的）

（快排空间复杂度是递归深度，即递归栈上需要花费的空间）

（冒泡排序指的是优化了的冒泡排序）

（选择排序每次选到数后是两数位置交换，所以不稳定）

（堆排序建堆时间为O(n)）

（快排操作时要先从后往前）

（快排时间复杂度最坏情况为初始序列有序或逆序时）

（希尔排序的每个子序列采用直接插入排序）

## Kmp算法时间复杂度

O (m+n) （包含求next数组的O(m)） 《南师数据结构》 p61

## 树

1. 满二叉树，完全二叉树
2. 高度为h的m叉树最多节点数目： （《教材》p88）
3. 具有n个节点的完全二叉树的高度：log(n+1)向上取整 （《教材》p90）
4. 二叉树有n个节点，则有n+1个指针域为空。 （《教材》p93）
5. 树的前根遍历等价于对应二叉树的前序遍历；

树的后根遍历等价于对应二叉树的中序遍历。

1. 树的四种存储结构（《教材》p113）
2. 树的孩子兄弟表示法根节点无右子树。

对于森林，第二棵树接在第一棵树根节点右子树上。

1. 二叉搜索树就是二叉排序树
2. 判断是否存在负权回路或正权回路：bellman-ford算法
3. B-树（即B树）、B+树、B\*树：在B+树的非根和非叶子结点再增加指向兄弟的指针

## 图

生成树：连通图的生成树是具有图全部节点的极小联通子图。n个顶点的生成树必有n-1条边。

网：带权图

连通图

强连通图：在有向图中，每一对节点都存在相互到达的路径。

强连通分量：非强连通图的极大强连通子图

简单路径：路径上经过的各顶点均不互相重复

## 二叉排序树的删除，AVL（平衡二叉查找树）的插入，B树的插入和删除

（略）

B树的插入删除都是朝根方向进行的，因为叶结点要在同一层上，不能向叶结点进行。

## 什么是红黑树，用来干什么的

红黑树是一种自平衡二叉查找树。最坏情况下，可以在O(log n)时间内做查找，插入和删除。

性质1. 节点是红色或黑色。

性质2. 根节点是黑色。

性质3. 每个叶节点（NIL节点，空节点）是黑色的。

性质4. 每个红色节点的两个子节点都是黑色。(从每个叶子到根的所有路径上不能有两个连续的红色节点)

性质5. 任一节点到其后代叶结点的所有路径都包含相同数目的黑色节点。

红黑树：广泛用在C++的STL中

B/B+树：用在磁盘文件组织和数据库索引。（B树是平衡的多路查找树）

## 查找20英里以内所有的餐厅，地图数据如何存储

使用R树，起码要知道R树的搜索，插入

http://blog.csdn.net/zhouxuguang236/article/details/7898272

## 图论知识

边 - 弧（有向边）

链 – 路

圈 – 回路

## 欧拉图，哈密尔顿图

欧拉提出“哥尼斯堡七桥问题”

## Hash函数及处理冲突的方法

（《王道 p243》或《南师教材 p192》）

hash函数：

1. 直接地址法 Hash(key)=a\*key+b
2. 除留余数法
3. 数字分析法
4. 平方取中法
5. 折叠法

处理冲突：

1. 开放地址法：线性探测法，平方探测法，再散列法（双Hash）

所谓开放地址法，指的是可存放新表项的空闲地址即向它的同义词开放，又向非同义词开放。

1. 链地址法（拉链法）

# 操作系统

主要包含1）进程2）内存3）文件4）I/O部分

可以根据《王道》目录进行复习。

## 进程调度

（《王道》p48）

1）FCFS 2）SJF 3）优先级调度算法 4）HRRF 5）RR 6）多级反馈队列调度算法

注：SJF是非剥夺的，而SRTF是剥夺的

## 页面置换

（《王道》p172）

1）OPT 2）FIFO 3）LRU 4）CLOCK

FIFO可能出现Belady异常。

## 磁盘调度

（《王道》p235）

1）FCFS 2）SSTF 3）电梯算法(SCAN) 4）C-SCAN

## 并行和并发区别

并行是指两个或多个事件在同一时刻发生。

并发是指两个或多个事件在同一时间间隔内发生。

## 进程线程区别，目的

区别：

A．进程是资源拥有的基本单位；线程是独立调度的基本单位

B．线程属于进程：

同一个进程中的线程切换不会引起进程切换；不同进程中的线程切换回引起进程切换。

一个程序至少有一个进程,一个进程至少有一个线程。

进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响；但线程之间没有单独的地址空间，一个线程死掉就等于整个进程死掉。

E．系统开销：

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，开销大于线程。

在进程切换时，涉及到当前进程CPU环境的保存及新被调度运行进程的CPU环境的设置，而线程切换时只需保存和设置少量的寄存器内容，不涉及存储器管理方面的操作。

（《王道》p31）

目的：

引入进程是为了使多道程序并发执行，以提高资源利用率和系统吞吐量

引入线程是为了减少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

## 线程

http://wenku.baidu.com/link?url=SLwbia9yan5NJOPbeHc0Z4tOPat9x8e5T93HgshrQqOZZV52rrAMw653XroTHw8pbIexQ1YZPmC7wHhk7Zcavnd1ga3pECDmz\_o6OlPSs7m

线程特点：

1.线程都具有一个ID

2.线程具有自己的安全属性

3.每个线程都有自己的内存栈

4.每个线程都具有自己的寄存器信息

创建线程的方法与区别（针对C/C++/MFC） ：

1、CreateThread( ) ——WIN32 API函数

2、\_beginthreadex( ) ——MS对C Runtime库的扩展SDK函数

3、AfxBeginThread( ) ——MFC中线程创建的MFC函数

## 作业，管程

（《程序员面试宝典》P253）

作业：用户在一次解题或一个事务处理过程中要求计算机系统所做工作的集合，包括用户程序、所需要的数据及控制命令等。作业是由一系列有序的步骤组成的。

管程：由一组数据以及定义在这组数据之上的对这组数据操作组成的软件模块。

## 目态 和 管态

大多数计算机系统将CPU执行状态分为目态与管态。CPU的状态属于程序状态字PSW的一位。CPU交替执行操作系统程序和用户程序。

管态 又叫特权态，系统态或核心态。CPU在管态下可以执行指令系统的全集。通常，操作系统在管态下运行。

目态 又叫常态或用户态。机器处于目态时，程序只能执行非特权指令。用户程序只能在目态下运行，如果用户程序在目态下执行特权指令，硬件将发生中断，由操作系统获得控制，特权指令执行被禁止，这样可以防止用户程序有意或无意的破坏系统。

从目态转换为管态的唯一途径是中断。

从管态到目态可以通过修改程序状态字来实现，这将伴随这由操作系统程序到用户程序的转换。

## 进程同步与互斥的共同点和不同点

互斥：是指某一资源同时只允许一个访问者对其进行访问，具有唯一性和排它性。但互斥无法限制访问者对资源的访问顺序，即访问是无序的。是一种竞争关系。

同步：是指在互斥的基础上（大多数情况），通过其它机制实现访问者对资源的有序访问。是一种协作关系。

## 进程/线程间同步机制

临界区（Critical Section）、互斥量（Mutex）、信号量（Semaphore）、事件（Event）

经典问题：生产者-消费者，读者-写者

1、临界区：通过对多线程的串行化来访问公共资源或一段代码，速度快，适合控制数据访问。在任意时刻只允许一个线程对共享资源进行访问，如果有多个线程试图访问公共资源，那么在有一个线程进入后，其他试图访问公共资源的线程将被挂起，并一直等到进入临界区的线程离开，临界区在被释放后，其他线程才可以抢占。

2、互斥量：互斥与临界区很相似，但是使用时相对复杂一些（互斥量为内核对象），不仅可以在同一应用程序的线程间实现同步，还可以在不同的进程间实现同步，从而实现资源的安全共享。

3、信号量：它允许多个线程在同一时刻访问同一资源，但是需要限制在同一时刻访问此资源的最大线程数目

PV操作及信号量的概念都是由荷兰科学家E.W.Dijkstra提出的。信号量S是一个整数，S大于等于零时代表可供并发进程使用的资源实体数，但S小于零时则表示正在等待使用共享资源的进程数。

4、事件：通过通知操作的方式来保持线程的同步，还可以方便实现对多个线程的优先级比较的操作。

http://www.jianshu.com/p/9218692cb209

http://blog.sina.com.cn/s/blog\_6b3a83c901019ajg.html

## 进程间通信的几种方法

（《王道-操作系统》P30）

（《教材》p190，较为详细，新美大校招面试时就问的很深入）

1）共享内存

在通信的进程之间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写/读操作实现进程间的消息交换。

2）消息传递

1.直接通信方式：发送进程直接把消息发送给接收进程，并把消息挂在接收进程的消息缓冲队列上。

2.间接通信方式：发送进程先把消息发送到信箱，接收进程从信箱中取得消息。

3）管道通信

管道的实质是一个共享文件，又名pipe文件。发送进程把数据送人管道，接收进程从管道中接收数据。

4）信号量( semophore )

进程同步也有信号量

5）信号( signal )

通过发送指定信号来通知进程某个异步事件发生

6）套接字( socket )

与其他通信方式主要的区别是：它可以实现不同主机间的进程通信

全局变量不是进程间通信的方法，因为每个进程都有自己独立的进程空间

共享内存比管道和消息队列效率高的原因：

http://www.lxway.com/905511464.htm

共享内存区是最快的可用IPC形式，一旦这样的内存区映射到共享它的进程的地址空间，这些进程间数据的传递就不再通过执行任何进入内核的系统调用来传递彼此的数据，节省了时间。

## 子进程自父进程继承什么或未继承什么

http://www.cnblogs.com/hnrainll/archive/2011/05/11/2043472.html

（以下只是部分）

继承到：

* 环境(environment)
* 堆栈
* 内存
* 打开文件的描述符
* 信号(signal)控制设定
* nice值 (译者注：nice值由nice函数设定，该值表示进程的优先级， 数值越小，优先级越高)
* 当前工作目录
* 文件方式创建屏蔽字(file mode creation mask (umask)) (译者注：译文取自《高级编程》，指：创建新文件的缺省屏蔽字)

子进程所独有：

* 进程号
* 自己的文件描述符和目录流的拷贝(译者注： 目录流由opendir函数创建，因其为顺序读取，顾称“目录流”)
* 在tms结构中的系统时间(译者注：tms结构可由times函数获得， 它保存四个数据用于记录进程使用中央处理器 (CPU：Central Processing Unit)的时间，包括：用户时间，系统时间， 用户各子进程合计时间，系统各子进程合计时间)
* 不继承由timer\_create函数创建的计时器
* 不继承异步输入和输出

## 子线程继承主线程的属性

http://blog.csdn.net/yusiguyuan/article/details/39432377

* 进程中的所有信息对该进程的所有线程都是共享的；
* 可执行的程序文本；
* 程序的全局内存；
* 堆内存；
* 栈；
* 文件描述符；
* 信号的处理是进程中所有线程共享的

## 孤儿进程和僵尸进程

（http://www.cnblogs.com/Anker/p/3271773.html）

孤儿和僵尸不要搞混！

孤儿进程：一个父进程退出，而它的一个或多个子进程还在运行，那么那些子进程将成为孤儿进程。孤儿进程将被init进程(进程号为1)所收养，并由init进程对它们完成状态收集工作。

僵尸进程：一个进程使用fork创建子进程，如果子进程退出，而父进程并没有调用wait或waitpid获取子进程的状态信息，那么子进程的进程描述符仍然保存在系统中。

一个进程在调用exit命令结束自己的生命的时候，其实它并没有真正的被销毁， 而是留下一个称为僵尸进程（Zombie）的数据结构。

## 死锁的四个必要条件

（《王道》p109）

互斥；不剥夺；请求等待；循环等待

死锁预防：破坏四个必要条件之一

死锁避免：银行家算法

## 冯诺依曼体系结构

1）五大部件组成

2）采用存储程序方式，指令和数据不加区别混合存储在同一个存储器中

3）指令和数据均以二进制代码表示

4）指令由操作码和地址码组成

5）指令在存储器中按其执行顺序存放

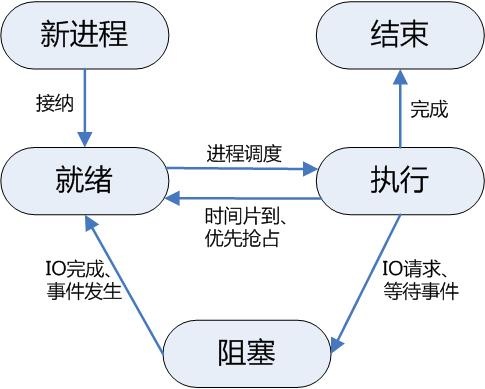
6）早期冯诺依曼机以运算器为中心，I/O设备与存储器间的数据传送都要经过运算器

## 进程的状态与转换：三态五态七态

运行，阻塞，就绪

运行，阻塞，就绪，创建，终止

创建，运行，静止就绪，活动就绪，静止阻塞，活动阻塞，结束



需要注意，运行态可以直接到就绪态。

## I/O控制方式

1）程序直接控制方式（程序查询方式）

2）中断驱动方式

3）DMA

4）通道控制方式

## 生产者消费者，读者写者

（《王道》p76）注意两者间区别

## 内存非连续分配管理

分页，分段，段页 （《王道》p142）

## 常见并发并发一致性问题：丢失修改，不可重复读，读"脏"数据，幻影读

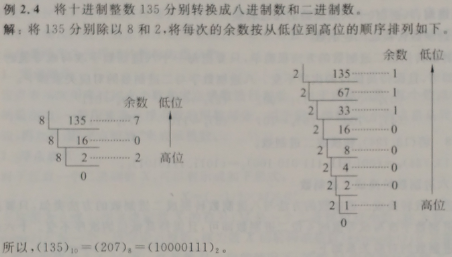
http://blog.163.com/magicc\_love/blog/static/185853662201111152200490/

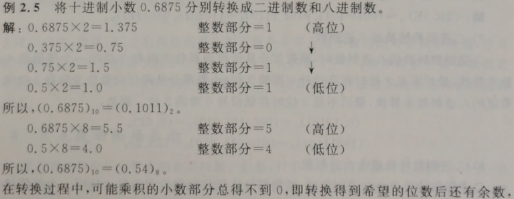
题目：http://www.nowcoder.com/test/question/done?tid=4902475&qid=36462

# 计组

## 十进制数转为R进制数

《计算机组成与系统结构》p29





## 格雷码

任何两个相邻的代码只有一个二进位的状态不同。

# 网络

可以根据《王道》目录进行复习。

## OSI参考模型，Tcp/IP模型

物理层，数据链路层，网络层，传输层，会话层，表示层，应用层

网络及接口层，网际层，传输层，应用层

各层功能:

物理层：利用传输介质为数据链路层提供物理连接，实现比特流的透明传输。

数据链路层：通过各种控制协议，将有差错的物理信道变为无差错的、能可靠传输数据帧的数据链路。

网络层：对分组进行路由选择，为通过通信子网选择最适当的路径。

传输层：端到端的数据传输

会话层：建立同步

表示层：数据压缩、加密和解密

应用层：采用不同的应用协议来解决不同类型的应用要求

http 属于（应用层）协议， ICMP 属于（网络层）协议。

## 在公司局域网上 ping www.taobao.com 没有涉及到的网络协议是（C）

A 、ARP B 、DNS C 、TCP D 、ICMP

DNS 是将域名 www.taobao.com 映射成主机的 IP 地址， ARP 是将 IP 地址映射成物理地址， ICMP 是报文控制协议，由路由器发送给执行 ping 命令的主机

## 当你在浏览器地址栏输入一个URL后回车，将会发生的事情

http://blog.csdn.net/wuhenliushui/article/details/20038819/

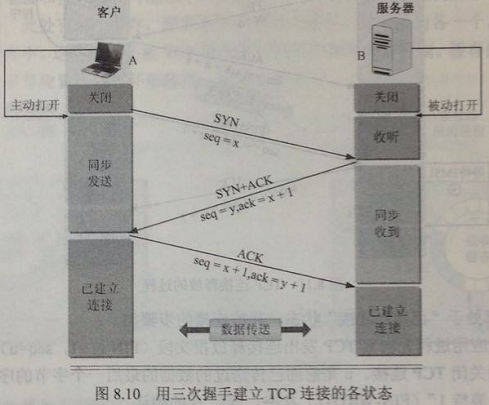
https://www.zhihu.com/question/20513729/answer/96332435

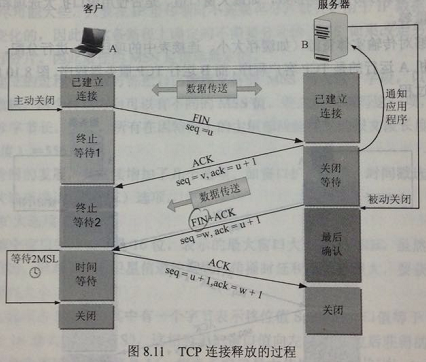
（《王道》p254）

1. 浏览器分析URL
2. 尝试从Host文件中获取URL的IP地址。
3. 向DNS请求解析URL的IP地址。
4. 浏览器缓存
5. 操作系统缓存
6. 路由器缓存
7. 本地域名服务器缓存。可以是ISP(InternetServiceProvider，互联网服务提供商)，若主机设置了DNS，按设置来、
8. 递归或迭代式依次查询根域名服务器，顶级域名服务器，授权域名服务器。其中从ISPDNS的配置文件里面读取13个根域名服务器的地址，向其中一台发起请求。
9. 与web服务器建立TCP连接（3次握手），其中还涉及到IP协议、ARP协议和OSPF协议。
10. 发送一个HTTP请求：GET
11. 收到服务器响应
12. 释放TCP连接（HTTP/1.1默认模式是使用流水线的持久连接《王道》p255）
13. 浏览器解析并显示网页

## 三次握手，四次交互

《南师教材 p221》 要知道每次的序号和确认号。





## 为何要“三次握手”？

（《王道》p229 6）A向B发出连接请求，第一个链接在网络中长时间滞留，A认为报文丢失。于是重传连接请求，B收到后建立了连接，数据传输完毕双方断开连接。此时之前滞留的连接到达B，若是“二次握手”，B会建立连接，并等待A传输数据。而“三次握手”，B会向A返回确认报文，A不予理睬，建立连接失败。

## DDos攻击

发送链接请求后，客户端不向服务器端发送最后一个数据包。

## 大量TIME\_WAIT或CLOSE\_WAIT原因及解决方法

（见TCP连接释放过程）

TIME\_WAIT（时间等待）是主动关闭连接的一方保持的状态，需等待2MSL后，TCP连接才被释放，进入关闭状态。解决方法：缩短等待时间。

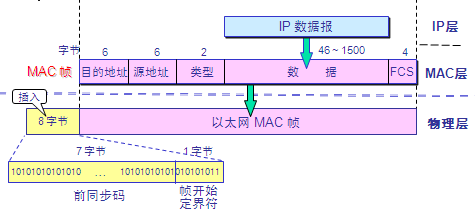
为何要2MSL，即为何要“四次交互”? 1.假想网络是不可靠的，你无法保证你最后发送的ACK报文会一定被对方收到。2. 通过一个2MSL TIME\_WAIT状态，确保所有的lost duplicate都会消失掉，避免对新连接造成错误。

CLOSE\_WAIT（关闭等待）是被动关闭连接的一方保持的状态，存在大量CLOSE\_WAIT的原因可能是忘记关闭连接，而一直等待关闭连接，这个是代码没写对造成的。

## MAC帧，IP数据报，TCP报文，HTTP请求报文 的格式

要熟悉每一部分含义。

MAC帧 《王道单科》p100



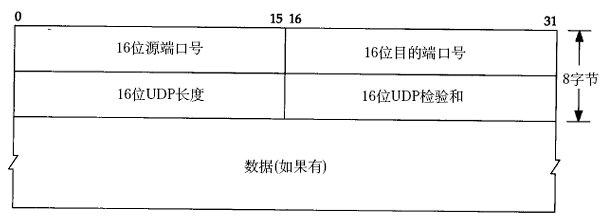
《王道单科》p138



《王道单科》p207



《王道单科》p201



## TCP和UDP区别

UDP：

是无连接的。

使用尽最大努力交付，即不保证可靠交付，同时也不使用拥塞控制。

是面向报文的（SOCK\_DGRAM）。UDP 没有拥塞控制，很适合多媒体通信的要求。

首部开销小，只有 8 个字节。

没有拥塞控制和流量控制。

TCP ：

是面向连接的。

TCP 提供可靠交付的服务。重复的报文会被检测出来，丢失重传，按序交给目的进程。

TCP是面向字节流（SOCK\_STREAM）。

TCP 提供全双工通信。

## 常用端口号

（TCP）FTP：20(数据)/21(控制) TELNET：23 SMTP：25 HTTP：80

（UDP）DNS：53 SNMP DHCP

（IP）ICMP(ping工作在应用层，tracert工作在网络层)

DHCP采用的是UDP《南师教材 p268》

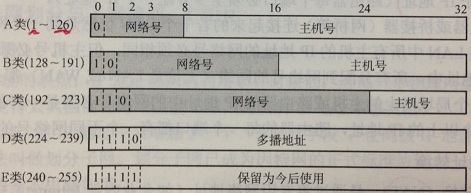
## OSI参考模型每一层的常用协议和硬件

见信纸笔记

## IPV6 地址包含了多少位

128

## IP地址分类



IP地址={网络号，子网号，主机号}

## socket编程

TCP和UDP的过程！

复习《TCP/IP协议与网络编程》

结合以前写的代码

## socket上调用recv函数，返回值为0表示

连接结束。

如果是发生错误，则返回SOCKET\_ERROR，应用程序通过WSAGetLastError()获取相应错误代码。

## cookie 和session 的区别

1、cookie数据存放在客户的浏览器上，session数据放在服务器上。

2、cookie不是很安全，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗，考虑到安全应当使用session。

3、session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能。考虑到减轻服务器性能方面，应当使用COOKIE。

4、单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。

## http请求报文中方法Get和Post区别

根据HTTP规范，GET用于信息获取；POST表示可能修改变服务器上的资源的请求。

Get使用URL传参，因为URL长度的限制而对传输数据有限制，向服务器发送大量数据时宜采用Post；且因为在URL中可以看到而安全性较低。Post是通过表单传送数据。

## HTTP协议状态码

1×× 　　提供信息

2×× 　　请求成功地接收

3×× 　　重定向

4×× 　　请求错误（客户错误）

5×× 　　服务器错误

其中：

200 OK 服务器已成功处理了请求

403 Forbidden 服务器拒绝请求

404 Not Found 服务器找不到任何匹配Request-URI的资源

## HTTPS和HTTP的概念和区别

HTTPS（全称：Hypertext Transfer Protocol over Secure Socket Layer），是以安全为目标的HTTP通道，简单讲是HTTP的安全版。即HTTP下加入SSL层。（http是明文传输）

https协议需要到ca申请证书，一般免费证书很少，需要交费。

http和https使用的是完全不同的连接方式用的端口也不一样,前者是80,后者是443。

## 其他

NAT内部地址和公用地址间的转换（《王道》p143）

DHCP，DNS

SNMP

CDN（Content Delivery Network，内容分发网络）

# 数据库

复习《数据库技术与应用》

## ACID，具体说明

数据库事务正确执行的四个基本要素的缩写。原子性(Atomicity）、一致性（Consistency）、隔离性（Isolation）、持久性（Durability）。

具体说明：（ 《数据库系统概论》王珊、萨师煊 p279）

原子性：事务中的所有操作要么都做，要么都不做，是不可分的原子工作

一致性：事务必须始终保持系统处于一致的状态，不管在任何给定的时间并发事务有多少。如果数据库系统运行中发生故障，有些事务尚未完成就被迫中断，这些未完成事务对数据库的修改有一部分已经写入数据库，这时数据库就处于不确定状态。一致性和原子性是密切相关的。

隔离性：一个事务的执行不能被其他事务打扰

持久性：事务一旦提交，对数据的改变就应该是永久的

## 模式

模式、外模式和内模式

## 将变量count值赋值为1

DECLARE @count --局部变量@开头，全局变量@@开头

SELECT @count=1

（《数据库技术与应用》p81）

## SQL语句

更新

## 主键，外键，唯一性约束

（《数据库技术与应用》p289）

主键：不能为NULL

外键：其他表中要为主关键字。当定义主关键字的表中更新列值时，关联的外关键字列也要相应更新。

唯一性（unique）：可以有NULL

## 视图

（略）

## 簇索引，非簇索引，唯一索引

索引是B树或B+树结构。

簇索引对表的数据按索引值排序，与表的数据存放在一起。

非簇索引与表的数据分离，叶结点中存储组成非簇索引的关键字的值和行定位器。

一个表只能建立一个簇索引，但能多个非簇索引。

唯一索引确保数据唯一性，只能有一个NULL值。

## 触发器（for，after，instead of）

（略）

## 连接查询

内连接（INNER JOIN）：查询结果仅包含两连接表中彼此相对应的数据记录

左外连接：查询结果包含左侧源表的全部数据记录，但仅包含右侧源表中与左表对应的记录

右外链接

完全外连接：查询结果包含两侧源表中所有相对应以及不对应的数据记录

自连接

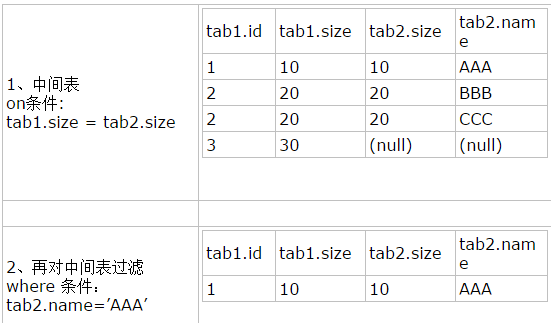
交叉连接：返回两个表的笛卡尔积

## join on 和where的执行顺序

两者区别见（《数据库技术与应用》p251）

数据库在通过连接两张或多张表来返回记录时，都会生成一张中间的临时表，然后再将这张临时表返回给用户。

select \* form tab1 left join tab2 on (tab1.size = tab2.size) where tab2.name=’AAA’



## 有哪些方式可以加快查询？

建立索引，其他…

## 范式

（《《数据库技术与应用》p312）

目前关系数据库有六种范式：第一范式（1NF）、第二范式（2NF）、第三范式（3NF）、巴斯-科德范式（BCNF）、第四范式(4NF）和第五范式（5NF，又称完美范式）。满足最低要求的范式是第一范式（1NF）。在第一范式的基础上进一步满足更多规范要求的称为第二范式（2NF），其余范式以次类推。一般说来，数据库只需满足第三范式(3NF）就行了。

1NF：每个属性都不可再分，是关系模式应具备的最起码的条件

2NF：非主属性完全依赖于主键（而不能只依赖于主键的一部分，不依赖于另一部分）[消除部分子函数依赖]

（减轻了1NF存在的插入异常、删除异常、数据冗余度大和修改复杂等问题）

3NF：非主属性不依赖于其它非主属性[消除传递依赖]

BCNF：在1NF基础上，任何非主属性不能对主键子集依赖[在3NF基础上消除对主码子集的依赖]

4NF：要求把同一表内的多对多关系删除。

5NF：从最终结构重新建立原始结构。

## 关系数据库与非关系数据库

关系数据库：

Oracle，Sybase，DB2（IBM产品），SQL Server

PostgreSQL：加州大学伯克利分校计算机系开发

非关系数据库：

MongoDB（分布式文档存储数据库）：是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。

Redis ：key-value存储系统

HBase：分布式的、面向列的开源数据库

## MySQL

（暂缺）

sqlserver中的top在mysql中为limit

## MySQL实现的四种通信协议

TCP/IP，Socket，共享内存，命名管道

（题目：http://www.nowcoder.com/test/question/done?tid=4902475&qid=36463#summary）

## 数据库的优化

考虑：

* 1. SQL语句以及索引的优化
  2. 表结构的优化
  3. 系统配置
  4. 硬件

# 软件工程

## 黑盒测试，白盒测试具体做法

黑盒测试：在测试时，把程序看作一个不能打开的黑盆子，在完全不考虑程序内部结构和内部特性的况下，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和正确的输出

白盒测试：按照程序内部的结构测试程序，通过测试来检测产品内部动作是否按照设计规格说明书的规定正常进行，检验程序中的每条通路是否都能按预定要求正确工作

## 软件开发模型

指软件开发全部过程、活动和任务的结构框架。软件开发包括需求、设计、编码和测试等阶段，有时也包括维护阶段。

## 软件过程模型

瀑布模型，增量模型，螺旋模型

## UML（统一建模语言）类图

<http://blog.csdn.net/tianhai110/article/details/6339565>

# 设计模式

## 设计模式分类

常考选择题。

设计模式分为三种类型，共23种。

**创建型：**

（sample factory 简单工厂模式 不算在里面）

factory method 工厂模式

abstract factory 抽象工厂模式

singleton 单例模式

prototype 原型模式

builder 建造者模式

**结构型：**

adapter 适配器模式：将一个类的接口转换成客户希望的另外一个接口

bridge 桥接模式

composite 组合模式

decorator 装饰模式

facade 外观模式

flyweight 享元模式

proxy 代理模式

**行为型：**

observer 观察者模式

template method 模版方法模式

command 命令模式

state 状态模式

chain of responsibility职责链模式

interpreter 解释器模式

mediator 中介者模式

visitor 访问者模式

strategy 策略模式

memento 备忘录模式

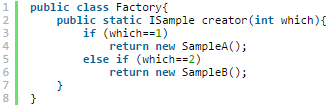
iterator 迭代器模式

## 设计模式举例

单例模式：保证一个类仅有一个实例，并提供一个访问它的全局访问点。

工厂模式：

http://blog.csdn.net/lingfengtengfei/article/details/12374469 ？



简单工厂、工厂方法、抽象工厂之小结与区别：

http://blog.csdn.net/superbeck/article/details/4446177

观察者模式：定义了一种一对多的依赖关系，让多个观察者对象同时监听某一个主题对象。这个主题对象在状态发生变化时，会通知所有观察者对象，使它们能够自动更新自己。

## 给出一种单例模式的实现

（http://blog.csdn.net/hackbuteer1/article/details/7460019）或《head first设计模式》

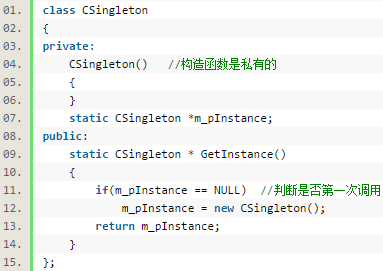
P173

比如，定义一个单例类，使用类的私有静态指针变量指向类的唯一实例，并用一个公有的静态方法获取该实例。

用户访问唯一实例的方法只有GetInstance()成员函数。如果不通过这个函数，任何创建实例的尝试都将失败，因为类的构造函数是私有的。

但仅这样还是不行的，还需要考虑实例的析构、线程安全（加锁，《head first设计模式》

P180），见



使用：

CSingleton\* p1 = CSingleton :: GetInstance();

CSingleton\* p2 = p1->GetInstance();

# 算法

## P与NP，解释并举例

（参见CSDN博客）

NP问题（Nondeterministic Polynomial）不是非P类问题（Non-polynomial），NP问题是指可以在多项式的时间里验证一个解的问题。如Hamilton图、TSP、3SAT、背包问题。所有的P问题都是NP问题。（SAT问题：判定一个给定的合取范式的布尔公式是否是可满足的）

有些问题甚至根本不可能找到一个正确的算法来，这称之为“不可解问题”(Undecidable Decision Problem)。The Halting Problem（停机问题）就是一个著名的不可解问题。

## 动态规划，分治

动态规划:把多阶段过程转化为一系列单阶段问题，利用各阶段之间的关系，逐个求解。

与分治法不同的是，适合于用动态规划求解的问题，经分解得到子问题往往不是互相独立的。

## A\*

把到达节点的耗散g(n)和从该节点到目标节点的消耗h(n)结合起来对节点进行评价：f(n)=g(n)+h(n)

## 蚁群算法，遗传算法、模拟退火算法

蚁群算法：

http://baike.baidu.com/view/539346.htm

遗传算法：

http://wiki.mbalib.com/wiki/%E9%81%97%E4%BC%A0%E7%AE%97%E6%B3%95

模拟退火：

http://wiki.mbalib.com/wiki/%E6%A8%A1%E6%8B%9F%E9%80%80%E7%81%AB%E7%AE%97%E6%B3%95

## 筛法求素数

把从1开始的、某一范围内的正整数从小到大顺序排列， 1不是素数，首先把它筛掉。剩下的数中选择最小的素数，然后去掉它的倍数。依次类推，直到筛子为空时结束。

## 数组中查找两个数使和为S？三个数字呢？

看arr[i] + arr[j] 是否等于Sum，如果是，则结束；如果小于Sum，则i = i + 1；如果大于Sum，则 j = j - 1。

三个数字时查找两个数使和为Sum-arr[i]。（https://leetcode.com/problems/3sum/）

## 数组中出现次数超过一半的数字

（《剑指offer》 面试题29）

方法一：基于Partition

方法二：考虑每次删除两个不同的数。

具体实现：实际上可以通过计数来实现，而不是物理上真正的删除。 在遍历数组的时候保存两个值：一个是数组中的一个数字，一个是次数。当我们遍历到下一个数字的时候，如果下一个数字和我们之前保存的数字相同，则次数加1。如果下一个数字和我们之前保存的数字不同，则次数减1。如果次数为零，我们需要保存下一个数字，并把次数设为1。

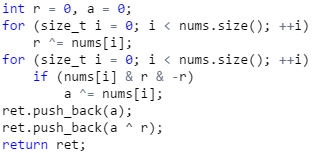
## 一个数组中所有数字都是成双成对出现的，只有一个数只出现一次，求这个数

异或和（为什么异或和能把相同的数消掉呢？从每一位来考虑，当这位上1出现两次就会变0）

关于命题逻辑的结合律、分配率见《离散数学》 p9。

当不成对的数有两个时，可以用以下代码：

（r&-r会保留最低位的1其他位置为0，而要求的两个数在这位是不同的，通过这样把这两个数分在不同的组）



## 如何快速找出一个32位整数的二进制表达里有多少个"1"？用关于"1"的个数的线性时间？

答案1（关于数字位数线性）：for(n=0; b; b >>= 1) if (b & 1) n++;

答案2（关于"1"的个数线性）：for(n=0; b; n++) b &= b-1;

## 判断给定的整数是否是一个2的幂。

(b & (b-1)) == 0

## 一个大小为N的数组，所有数都是不超过N-1的正整数。找出重复的那个数（假设只有一个）。

答案：计算数组中的所有数的和，再计算出从1到N-1的所有数的和，两者之差即为重复的那个数。

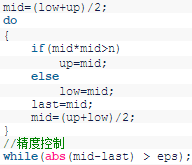
同理，一个大小为N的数组，所有数都是不超过N+1的正整数。找出没有出现过的那个数（假设只有一个）。计算数组中的所有数的和，再计算出从1到N+1的所有数的和，两者之差即为缺少的那个数。

扩展：如果是两个数呢？（《编程之美》p41 快速找出故障机器）

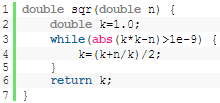
两个数要有两个方程，除了求和再添加个乘积或平方和。

## 开更号

1）二分查找



2）牛顿法http://www.nowamagic.net/librarys/veda/detail/2268



3）先用”平方根倒数速算法”求开根号倒数，再乘以要开更的数

## 判断是否有环

1. 用两个指针，一个每次递增一步，一个每次递增两步，如果有环的话两者必然重合
2. 对访问过的元素标记，遍历过程若遇到做过标记的元素说明有环 （《C专家编程》p274）

## 判断两个链表是否相交并找出交点

1）将链表节点指针分别存入stack，出栈判断是否相同

2）将第一个链表尾结点指向第二个链表头节点，判断是否有环（只能判断是否相交，无法找出交点）

3）两个链表的最后一个节点是否相同（判断出是否相交），让长度较长的链表指针向后移动|len1 - len2|，然后在开始遍历两个链表，判断节点是否相同即可。

*面试官问这个问题，如果回答了方法1，面试官可能追问空间复杂度为O(1)的方法，这时可以介绍方法3。*

扩展：如果链表可能有环，则如何判断两个链表是否相交？

链表1步长为1，链表2步长为2，如果有环且相交则肯定相遇

## 两个排好序的字符串（如URL）数组求公共字符串

（百度3面面试题，如果给的是整数的话应该能立即想到解法，但给的是字符串，刚开始时蒙了。）两个数组中指向的当前字符串比较后移动下标。（string可以直接用比较运算符比较）

## 两个排好序的数组，求中位数，要求时间复杂度O(log(m+n))

（leetcode 4 Median of Two Sorted Arrays）

比较A[k/2-1]和B[k/2-1]两个元素。

## 求n个实数在实轴向量上2个数之间的最大差值

1. 找到n个数据中最大和最小数据max和min。
2. 用n-2个点等分区间[min, max]，除最大最小数据max和min以外的n-2个数据放入n-1个桶中。
3. 由抽屉原理可知至少有一个桶是空的，最大间隙定超过一个桶的大小，所以最大间隙不会在同一桶中出现，一定是某个桶的上界和气候某个桶的下界之间隙。

## 最大子矩阵

分解为最大子段和。（见我的CSDN博客）

## 一棵树中两个结点的最近公共祖先

http://blog.163.com/zhaohai\_1988/blog/static/209510085201263195947966/

LCA：Least Common Ancestors（最近公共祖先）

1. （如果这棵树只有父节点指向子结点）先获取根节点到节点u和v的路径，这样可以转化为“判断两个单链表是否相交”问题（《剑指offer》面试题37）。
2. （树是无向图形式，如矩阵存储）DFS搜索u到v的路径，然后判断祖先节点
3. Tarjan算法？

## Catalan数

（ 《王道 程序员求职宝》p211 ，百度百科）

**由来：**



前几项为（从0开始）: 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862。

**公式：**（两个公式都得记住，微软12年实习生、腾讯12年实习生、阿里都出过相关题目）





**应用：**

括号匹配问题（n对括号有多少种可能排列）分析出递推式：

1. 第一组括号中没有其他括号，()…，第一组括号右边剩下n-1组括号
2. 第一组括号中有一对括号，(())…，第一组括号右边剩下n-2组括号
3. 第一组括号中有n-1对括号，(…)，第一组括号右边没有其他括号

根据第一组括号划分出n中情况，从上面可以得到h(n)公式。

（以下问题推导都是从递推式角度来想。遇到一个类似于括号匹配的问题，就该想到卡特兰数）

1.括号化问题。

矩阵链乘： P=a1×a2×a3×……×an，依据乘法结合律，不改变其顺序，只用括号表示成对的乘积，试问有几种括号化的方案？(h(n)种)

2.出栈次序问题。

一个栈(无穷大)的进栈序列为1,2,3,..n，有多少个不同的出栈序列?

类似：有2n个人排成一行进入剧场。入场费5元。其中只有n个人有一张5元钞票，另外n人只有10元钞票，剧院无其它钞票，问有多少中方法使得只要有10元的人买票，售票处就有5元的钞票找零？

类似：在图书馆一共6个人在排队，3个还《面试宝典》一书，3个在借《面试宝典》一书，图书馆此时没有了面试宝典了，求他们排队的总数？（2012腾讯实习招聘笔试题）C3=5，所以总数为5\*3!\*3!=180.

3.将多边行划分为三角形问题。

将一个凸N+2多边形区域分成三角形区域的方法数?

类似：一位大城市的律师在她住所以北n个街区和以东n个街区处工作。每天她走2n个街区去上班。如果她从不穿越（但可以碰到）从家到办公室的对角线，那么有多少条可能的道路？

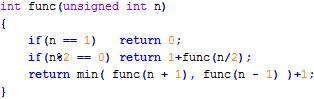
类似：在圆上选择2n个点,将这些点成对连接起来使得所得到的n条线段不相交的方法数?

4.给顶节点组成二叉树的问题。

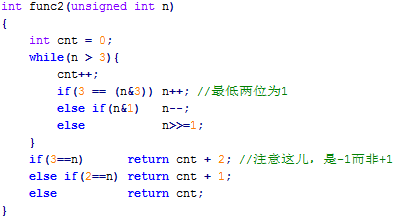
给定N个节点，能构成多少种不同的二叉树？（第一次根据根节点所在位置划分出n种情况）

## 对一个正整数n，算得到1需要的最少操作次数：如果n为偶数，将其除以2；如果n为奇数，可以加1或减1

方法一：递归



方法二：位运算，以下为非递归写法，也可写出递归形式。其中判断最低两位是否为一不用判断两次，只要&3即可，或是用n % 4。



## 设计DNS服务器中cache的数据结构

要求设计一个DNS的Cache结构，要求能够满足每秒5000以上的查询，满足IP数据的快速插入，查询的速度要快。（题目还给出了一系列的数据，比如：站点数总共为5000万，IP地址有1000万，等等）

DNS服务器实现域名到IP地址的转换。每个域名的平均长度为25个字节（估计值），每个IP为4个字节，所以Cache的每个条目需要大概30个字节。总共50M个条目，所以需要1.5G个字节的空间。可以放置在内存中。（考虑到每秒5000次操作的限制，也只能放在内存中。）可以考虑的数据结构包括hash\_map，字典树，红黑树等等。

# 大数题

* 1. 分而治之：Hash，桶(bucket)排序
  2. Bitmap 使用STL bitset或p+(i/8)|(0×01<<(i%8))
  3. Trie树
  4. 外排序
  5. 倒排索引
  6. MapReduce
  7. Bloom Filter

注意败者树（多路归并是二叉树，不是多叉树！）和堆的区别

1. 败者树是将归并段挂在叶结点上，并且非叶结点是“失败者”，而堆的数据放在整个堆上。
2. 败者树是从下往上维护，每上一层，只需要和败者节点比较“一次”即可。而堆在维护的时候是从上往下，每下一层，需要和左右子节点都比较，需要比较两次。
3. 败者树每一次维护必定需要从叶子节点一直走到根节点，不可能中间停止；而堆维护时，“有可能”会在中间的某个层停止，不需要继续往下。

## 求两个数组（URL等）交集

1）暴力解法——遍历

2）先排序，之后看166中方法

3）若数组内数字的范围是已知并且不是很大，建立新的数组统计重复的数字。

4）哈希分成小文件，再用hash\_set

5）把第一个数组存入set （查找要能立即想到用二分查找优化）

## 最大k个数

* 1. 快排思想
  2. 大数情况，小顶堆（或是红黑树multiset<int,less<int>> 见《剑指offer》p168）
  3. 大数情况，k若较大，先按大小划分区域，统计落到各个区域里的数的个数再处理。如：5亿个int找它们的中位数。第一次遍历根据int的前几位统计出每个区域个数，不用真的去hash分区域。
  4. 数据是存储在排序树中的，树的遍历（先右子树后父节点后左子树）

## 出现次数最多（有一个1G大小的一个文件，里面每一行是一个词，词的大小不超过16字节，内存限制大小是1M。返回频数最高的100个词。）

先hash分成几部分，（可以采用trie树/hash\_map等）将词及相应的次数存入文件，对每个文件中出现最多的那100个再拿出来归并（或是败者树）。

## 论坛有注册ID两亿个，日志文件中记有登陆时间和退出时间，统计一天中用户在线分布，取样粒度为秒

delta[86400]，每个整数对应这一秒的人数变化值，可能为正也可能为负。

online\_num[86400]，每个整数对应这一秒的论坛在线人数。

第n秒的人数online\_num[n]=online\_num[n-1]+delta[n]。

## 大量未排序的不重复的unsigned int的整数，判断某个数是否在里面

申请连续的2^32/8=512M的内存，用每一个bit对应一个unsigned int数字

（遇到大数题也要考虑下是否能装入内存）

对于Bitmap，有时也可以用2-Bitmap，00表示不存在，01表示出现一次，02表示出现多次。

## Bloom Filter

申请一个数组，初始化为0.定义k个不同的hash函数，每个函数都可以将集合中的元素映射到位数组的某一位。

不适合用于“零错误”场合。

# 编程题

复习《C语言笔记》 ，《OJ笔记》 ，Leetcode收藏的题，CSDN中ACM博客

## 复习Leetcode收藏的题

（无）

## 复习STL，string，字符串函数

见《C语言笔记》

## 复习文件读写，运算符重载，lamada表达式，函数指针

关于运算符重载，注意类的成员函数和友元函数时的区别，为何不能将流运算符作为成员函数重载

对于函数 void max(int a){ … }

函数指针的定义和使用：

void (\*p)(int)=max;

p(2);

(\*p)(2);

## 大整数加、减、乘、除、求模运算

略

## 递归：汉诺塔，八皇后

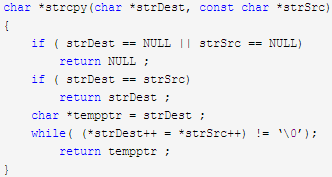
略

## 动态规划常见问题

背包问题：0-1背包，完全背包，多重背包 （前两个算法容易忘了，要经常复习）

最长递增子序列（LIS），最长公共子序列（LCS）

## C语言实现strcpy函数



类似题目有《剑指offer》 p24面试题1：赋值运算符

# 机器学习

前几个问题按照《统计学习方法》书上顺序。

## 产生式模型和判别式模型

（《统计学习方法》p18）

产生式：学习联合概率分布P(X,Y)，然后求出条件概率分布P(Y|X)。

之所以称为生成方法，是因为模型P(Y|X)= P(X,Y)/P(X) 表示了给定输入X产生输出Y的生成关系。

Naive Bayes，HMM

Sigmoidal Belief Networks,

Bayesian Networks

Markov Random Fields

Latent Dirichlet Allocation

判别式：直接学习决策函数f(X)或者条件概率分布P(Y|X)。

输入特征X，就直接预测出最可能的Y。

perceptron，k-nearest neighbor，decision tree，Logistic Regression，SVM，Boosting，CRF

Traditional Neural Networks

Linear Discriminant Analysis

Linear Regression

## 分类算法评价标准

正确率（accuracy）：accuracy = (TP+TN)/(P+N) ，P表示实际为正例的样本个数

精度（precision）：被分为正例的示例中实际为正例的比例，precision=TP/(TP+FP)

召回率（recall）： 有多少正例被分为正例，recall=TP/(TP+FN)=TP/P

F1测试值： （《添加学习方法》p19）

PR曲线：y轴和x轴分别为Precision和Recall

ROC曲线：y轴和x轴分别为TPR=TP/P（真正类率）和FPR=FP/N（假正类率）

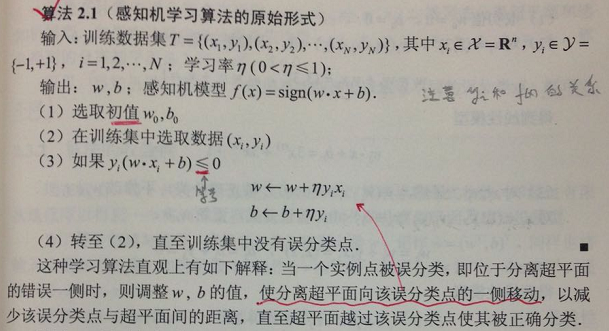
http://blog.csdn.net/hermito/article/details/51076728

ROC曲线下方的面积（Area Under the ROC Curve, AUC）提供了评价模型平均性能的另一种方法。如果模型是完美的，那么它的AUC = 1，如果模型是个简单的随机猜测模型，那么它的AUC = 0.5。

ROC通常用于衡量二分类器的好坏。（然而，当处理正负样本分布得极不均匀数据集，P-R曲线展示信息更具代表性。 ？）（实际环境中正负样本极不均衡，PR曲线无法很好反映出分类器性能，而ROC受此影响小。 ？）

## 感知机使用SGD学习算法

（统计学习方法 p29）



## k近邻（k-NN）

kd树的构造和搜索

局部敏感哈希求解KNN（http://blog.csdn.net/hermito/article/details/51682282?locationNum=1）

## NB公式及推导，先验概率、后验概率、联合概率、条件概率

（统计学习方法 p47）

先验概率( Prior probability )：先验指无需经验或先于经验获得的知识，是指根据以往经验和分析得到的概率

后验概率( posterior probability )：事情已经发生，要求这件事情发生的原因是由某个因素引起的可能性的大小

## Sigmoid函数

http://d.hiphotos.baidu.com/baike/s%3D99/sign=a46bd6f1dd33c895a27e9472d01340df/0df3d7ca7bcb0a4659502a5f6f63f6246b60af62.jpg

## 决策树

熵，条件熵，信息增益，信息增益比，基尼指数 的公式和含义

属性选择：

ID3 - 信息增益最大

C4.5 - 信息增益比最大

CART（分类与回归树）- 基尼指数最小

剪枝算法：比较剪去和不剪时的损失函数大小

## 逻辑斯谛回归及随机梯度下降求解

二项逻辑斯蒂回归公式：



SGD的变种：Maximum Likelihood, Logistic Regression,and Stochastic Gradient Training

## SVM目标和学习算法

间隔最大化

学习算法：《统计学习方法》p100

## SVM核函数

核函数K（kernel function）就是指K(x, y) = <f(x), f(y)>，其中x和y是n维的输入值，f(·) 是从n维到m维的映射（通常而言，m>>n）。<x, y>是x和y的内积（inner product）。

对于非线性问题，可将样本从原始空间映射到高维的特征空间，使得样本在这个空间线性可分。维度变大后，计算变多，使用核函数可以减少计算。

高斯核，对应的支持向量机是高斯径向基函数分类器。

http://img.my.csdn.net/uploads/201304/03/1364958259_8460.jpg

## 集成方法

《机器学习 周志华》第8章

序列化方法：AdaBoost（Boosting方法的代表）《统计学习方法》p138

并行化方法：Bagging，随机森林

## 如何选择分类算法

（网上也没有明确回答）

可以参考scikit-learn algorithm cheat-sheet。

如果是小训练集，高偏差/低方差的分类器（比如朴素贝叶斯）要比低偏差/高方差的分类器（比如k最近邻）具有优势，因为后者容易过拟合。然而随着训练集的增大，低偏差/高方差的分类器将开始具有优势（它们拥有更低的渐近误差），因为高偏差分类器对于提供准确模型不那么给力。

## 距离度量

Lp距离（统计学习方法 p38）

曼哈顿距离，欧式距离

马氏距离：

D_M(x) = \sqrt{(x - \mu)^T \Sigma^{-1} (x-\mu)}.\, 

## 过拟合和范数

过拟合（overfitting）是指数据模型在训练集里表现非常满意，但是一旦应用到真实业务实践时，效果不行。

发生overfitting 的主要原因是：（1）使用过于复杂的模型；（2）数据噪音，模型过分记住了噪音特征；（3）有限的训练数据；（4）建模时的“逻辑假设”到了模型应用时已经不能成立

解决：

1. 交叉验证（Cross-validation），为什么能防止过拟合？均值，方差。
2. 正则化，就是对最小化经验误差函数上加约束。

正则化就是把额外的约束或者惩罚项加到已有模型（损失函数）上，以防止过拟合并提高泛化能力。损失函数由原来的E(X,Y)变为E(X,Y)+alpha||w||，w是模型系数组成的向量（有些地方也叫参数parameter，coefficients），||·||一般是L1或者L2范数，alpha是一个可调的参数，控制着正则化的强度。当用在线性模型上时，L1正则化和L2正则化也称为Lasso和Ridge。（http://dataunion.org/14072.html）

（https://www.zhihu.com/question/20924039）采用L1 regularizer，它的优良性质是能产生稀疏性，导致 W 中许多项变成零。采用L2 regularizer，使得模型的解偏向于 norm（范数）较小的 W。

L0范数是指向量中非0的元素的个数。L0范数的最小化问题在实际应用中是NP难问题。因此很多情况下，L0优化问题就会被relaxe为更高维度的范数问题，如L1范数，L2范数最小化问题。

随机森林不易产生过拟合的原因：随机森林的随机性体现在每棵树的训练样本是随机的，树中每个节点的分裂属性也是随机选择的。两个随机性的引入，使得随机森林不容易陷入过拟合。

## K-means

（暂略）

（见 数据挖掘\Assignment 4）

（对比《Science》Clustering by fast search and find of density peaks 类簇中心被具有较低局部密度的邻居点包围，且与具有更高密度的任何点有相对较大的距离。）

## 常见聚类方法

《机器学习.周志华》第九章

1）原型：原型是指样本空间中具有代表性的点

1. k-means

2. LVQ（Learning Vector Quantization，学习向量量化）：假设样本带有类别标记。初始化原型向量，迭代优化。对任意样本，将被划入与其距离最近的原型向量所代表的簇中。

2）密度

DBSCAN（Density-Based Spatial Clustering of Applications with Noise）：先根据领域参数找出所有核心对象，以任一核心对象为出发点，找出其密度可达的样本生成聚类簇，直到所有核心对象都被访问过。

3）层次

AGNES（AGglomerative NESting）：先将数据集中每个样本看做一个初试聚类簇，每一步找出距离最近的两个聚类簇合并，直到达到预设的个数。（其中“距离”可以是多种距离）

--------------------------参考一些论文的分类分法------------------------------

1）层次

1. 聚合聚类：CURE算法

2. 分解聚类

2）划分

1.基于图论（构造最小生成树）

2.K-means

3）密度：DBSCAN

4）网格：STING

5）机器学习

1.神经网络：SOM

2.基于进化理论（模拟退火，遗传算法）

## LDA和PCA比较

两者都是为了在对原始数据降维之后进行分类。PCA是无监督的方式，它没有分类标签，降维之后需要采用K-Means或自组织映射网络等无监督的算法进行分类。LDA是有监督的方式，它先对训练数据进行降维，然后找出一个线性判别函数。

http://www.cnblogs.com/zhangchaoyang/articles/2644095.html

## 常见算法

最小二乘

BP网络：http://blog.csdn.net/hermito/article/details/51507028?locationNum=1

GBDT迭代决策树

Word2vec：利用深度学习的思想，可以通过训练，把对文本内容的处理简化为 K 维向量空间中的向量运算。

在线学习算法FTRL：http://www.cnblogs.com/EE-NovRain/p/3810737.html

## 分词方法

目前的分词方法归纳起来有3 类:

1） 基于语法和规则的分词法。其基本思想就是在分词的同时进行句法、语义分析, 利用句法信息和语义信息来进行词性标注, 以解决分词歧义现象。因为现有的语法知识、句法规则十分笼统、复杂, 基于语法和规则的分词法所能达到的精确度远远还不能令人满意, 目前这种分词系统还处在试验阶段。

2） 机械式分词法（即基于词典）。机械分词的原理是将文档中的字符串与词典中的词条进行逐一匹配, 如果词典中找到某个字符串, 则匹配成功, 可以切分, 否则不予切分。基于词典的机械分词法, 实现简单, 实用性强, 但机械分词法的最大的缺点就是词典的完备性不能得到保证。

3） 基于统计的方法。基于统计的分词法的基本原理是根据字符串在语料库中出现的统计频率来决定其是否构成词。词是字的组合, 相邻的字同时出现的次数越多, 就越有可能构成一个词。因此字与字相邻共现的频率或概率能够较好的反映它们成为词的可信度。

P(W)=P(W1) P(W2lW1) P(W3|W1W2)⋯P(Wk|W1,W2...Wk-1)

马尔可夫假设，N-Gram 《数学之美》p32

常用工具：jieba(python)，IKAnalyzer（java）

## Bag of words model (词袋模型)

在这种模型中，文本（段落或者文档）被看作是无序的词汇集合，忽略语法甚至是单词的顺序。

## tf\*idf

词频（term frequency，TF），某一个给定的词语在该文件中出现的频率。

逆向文件频率（inverse document frequency，IDF），可以由总文件数目除以包含该词语之文件的数目。

tfidf是用来衡量单词对文档的重要性的，用于提取关键词。之后的特征向量可以为关键词出现次数。

## 特征描述算子

**LBP**（Local Binary Pattern，局部二值模式）是一种用来描述图像局部纹理特征的算子；它具有旋转不变性和灰度不变性等显著的优点。

【从LBP 的定义可以看出，LBP 算子是灰度不变的，但却不是旋转不变的。Maenpaa等人又将 LBP 算子进行了扩展，提出了具有旋转不变性的LBP算子，将进制码按位旋转，取最小值。为了解决二进制模式过多的问题，提高统计性，Ojala提出了采用一种“等价模式”（Uniform Pattern）。 http://blog.csdn.net/xidianzhimeng/article/details/19634573 】

SIFT（Scale Invariant Feature Transform，尺度不变特征变换）对于图像的缩放、旋转和仿射变换等具有不变性。

SURF（Speed-Up Robust Features）

BRIEF（binary robust independent elementary features）是Calonder等在2010年提出来的，不具有旋转不变性。

ORB (Oriented FAST and Rotated BRIEF)是Rublee等在2011年提出来的，是建立在改进的FAST特征和改进的rBRIEF特征之上。改进了 BRIEF描述子对图像噪声敏感和不具备旋转不变性的缺点，但没有解决尺度不变性。

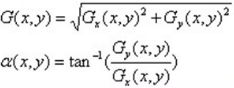
HOG本身不具有旋转不变性。

----------------------------------------------------------------------

**HOG**（Histogram of Oriented Gradient，方向梯度直方图）：对每个cell按梯度方向划分，做梯度大小的统计。（win – block - cell）

算法实现 http://blog.csdn.net/zouxy09/article/details/7929348





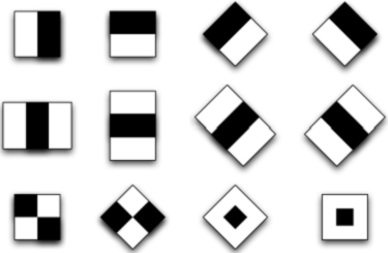
OpenCV调用 http://blog.csdn.net/hermito/article/details/48155557

----------------------------------------------------------------------

**SIFT**（Scale-invariant feature transform，尺度不变特征变换）

----------------------------------------------------------------------

**Haar-like**



人脸检测：AdaBoost using harr-like features in a cascade structure

## 设计行人检测系统

（360实习生一面问题）

方法一：HOG+SVM

方法二：（http://shijuanfeng.blogbus.com/logs/100675201.html）

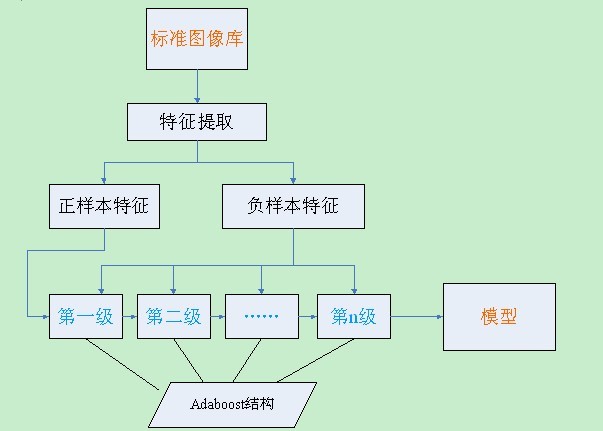
**离线训练过程：**

1.输入为标准图像库中的图像，对于正样本，直接提取特征；对于负样本，多尺度遍历整幅图像，进行特征提取。提取Harr，LBP，HOG三种特征并保存为积分图形式。

2.对于cascade的每一级进行如下操作：用Adaboost进行特征选择，选出最好的特征。

在前面的几级用Harr和LBP这两种计算较快的特征，后几级用HOG。

3.进行级数调整，直到达到要求的检测率和错误率。



**检测过程：**

1.输入图像，多尺度遍历整幅图像，进行特征提取：提取Harr，LBP，HOG三种特征并保存为积分图形式。

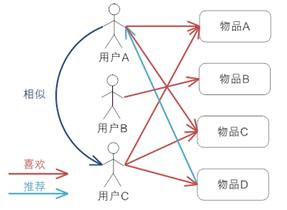
2.将所有子窗口输入训练得到的模型中，如下图，经cascade分类器后，根据检测结果即可标定行人区域，通过一些后处理（例如，合并临近的窗口），就可以对行人进行定位，达到检测的目的。

## 推荐系统

（https://www.ibm.com/developerworks/cn/web/1103\_zhaoct\_recommstudy1/）

基于user推荐，基于item推荐，协同过滤（user，item）

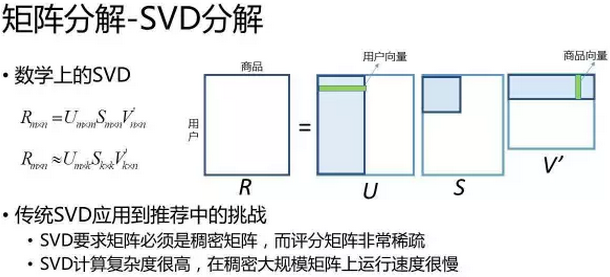
基于用户的协同过滤推荐的基本原理是，根据所有用户对物品或者信息的偏好，发现与当前用户口味和偏好相似的“邻居”用户群，在一般的应用中是采用计算“K- 邻居”的算法；然后，基于这 K 个邻居的历史偏好信息，为当前用户进行推荐。如下图：

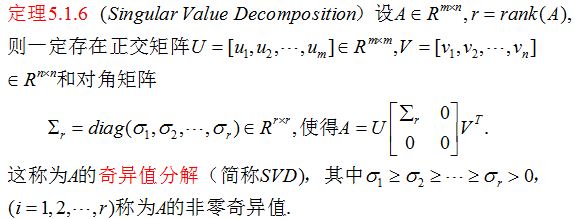


基于项目的协同过滤推荐的基本原理也是类似的，只是说它使用所有用户对物品或者信息的偏好，发现物品和物品之间的相似度。

常用推荐算法：

http://mp.weixin.qq.com/s?\_\_biz=MzIzOTU0NTQ0MA==&mid=2247483811&idx=1&sn=fc3ee4ddfc4a8d6014a4cd90cdb5983c&scene=0#rd



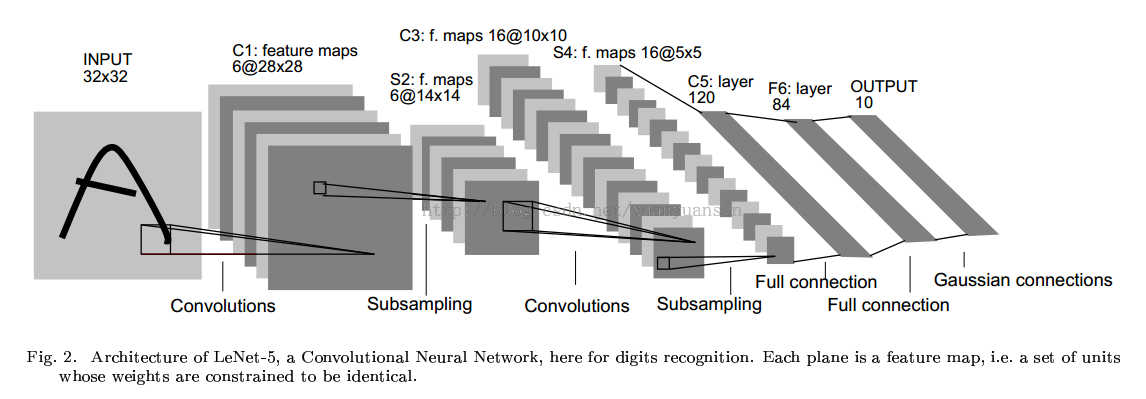


## 深度学习

LeNet5的基本结构：

《视觉机器学习》p181

http://blog.csdn.net/xuanyuansen/article/details/41800721



# 项目

## ZTE高层语义特征学习

LMNN算法 （《机器学习 周志华》 p241个人笔记）







 hinge loss

flandmark算法：

DPM（在深度学习前很火的目标检测算法），SO-SVM求解参数，SGD

属性举例不要举嘴巴张的例子，这个属性不适用于人脸识别

# 图像处理

## 滤波

平滑滤波器可用低通滤波器实现，目的是模糊和消除噪声。

锐化滤波器可用高通滤波器实现，目的是增强被模糊的细节边缘。

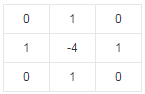
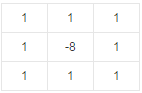
既要消除噪声又要保持细节可以使用中值滤波。（取中间值）

## 边缘检测方法

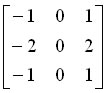
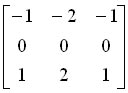
Roberts算子：

基于Roberts算子的边缘检测

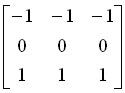
Laplacian算子：（4 邻域，8 邻域）

Sobel算子：

Prewitt算子：

Canny算子

# Linux

## Linux根目录下文件夹

/var 包含系统一般运行时要改变的数据。通常这些数据所在的目录的大小是要经常变化或扩充的。原来 /var 目录中有些内容是在 /usr 中的，但为了保持 /usr 目录的相对稳定，就把那些需要经常改变的目录放到 /var 中了

/etc 包含各种系统配置文件

/bin 二进制可执行命令

/sbin 系统管理命令

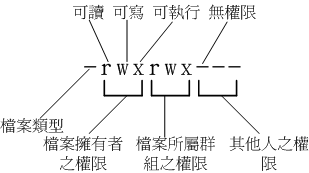
/mnt 挂载(mount)其他的文件系统

## /etc/目录 文件介绍

1. /etc /services 定义了TCP/IP服务和他们的端口分配
2. /etc/hosts 提供主机名到IP地址的对应关系
3. /etc/resolv.conf DNS服务的配置文件，例如：



## linux文件权限



文件类型部分（档案类型）：

-为：表示文件

d为：表示文件夹

l为：表示链接文件，可以理解为 windows中的快捷方式（link file）

b为：表示里面可以供存储周边设备

c为：表示里面为一次性读取装置

修改：

chmod 755 t.py

chmod u+x file 给file的属主增加执行权限

## 命令

grep命令是一种强大的文本搜索工具,它能使用正则表达式搜索文本,并把匹 配的行打印出来。grep全称是Global Regular Expression Print。

awk是一个强大的文本分析工具，相对于grep的查找，sed的编辑，awk在其对数据分析并生成报告时，显得尤为强大。简单来说awk就是把文件逐行的读入，以空格为默认分隔符将每行切片，切开的部分再进行各种分析处理。

ps：查看进程

top：查看各个进程的资源占用状况，，类似于Windows的任务管理器

free：单独查看内存使用情况

kill：中止一个进程

df：查看磁盘使用情况

## 哪个指令是将所有缓冲区的内容写入到磁盘？

sync，不是flush！

示例：[root@web ~]# sync

## 文本文件内容查看命令

cat：连接并显示一个或多个文件的有关信息

tac：从最后一行开始显示

more：类似cat，不过会以一页一页的显示方便使用者一页页阅读

less：比more更强大一点，支持各种命令，随便翻页、跳转、查找

head：只显示头几行

tail：只显示尾几行

示例：

将文件aaa中包含“root“的行的内容显示出来：cat aaa | grep “root”

## wc

是一个用于统计的工具，全拼为：Word Count(单词统计)

* -c 统计字节数。
* -l 统计行数。
* -m 统计字符数。这个标志不能与 -c 标志一起使用。
* -w 统计字数。一个字被定义为由空白、跳格或换行字符分隔的字符串。

统计prog.c函数中for语句通过的次数：grep “for” proc.c | wc -l

## 网络常用测试命令

1. ping
2. tracert 路由跟踪
3. telent 远程登录；网络测试

如telnet 114.212.80.220 80，测试80端口的web服务器是否正常工作

1. netstat 用于显示与IP、TCP、UDP和ICMP协议相关的统计数据，一般用于检验本机各端口的网络连接情况

## inode

(http://www.ruanyifeng.com/blog/2011/12/inode.html)

1）文件储存在硬盘上，硬盘的最小存储单位叫做"扇区"（Sector）。每个扇区储存512字节（相当于0.5KB）。

操作系统读取硬盘的时候，不会一个个扇区地读取，这样效率太低，而是一次性连续读取多个扇区，即一次性读取一个"块"（block）。这种由多个扇区组成的"块"，是文件存取的最小单位。"块"的大小，最常见的是4KB，即连续八个 sector组成一个 block。

文件数据都储存在"块"中，那么很显然，我们还必须找到一个地方储存文件的元信息，比如文件的创建者、文件的创建日期、文件的大小等等。这种储存文件元信息的区域就叫做inode，中文译名为"索引节点"。

2）一般情况下，文件名和inode号码是"一一对应"关系，每个inode号码对应一个文件名。但是，Unix/Linux系统允许，多个文件名指向同一个inode号码。

这意味着，可以用不同的文件名访问同样的内容；对文件内容进行修改，会影响到所有文件名；但是，删除一个文件名，不影响另一个文件名的访问。这种情况就被称为"硬链接"（hard link）。inode信息中有一项叫做"链接数"，记录指向该inode的文件名总数。

*ln命令可以创建硬链接：ln 源文件 目标文件*

3）除了硬链接以外，还有一种特殊情况。

文件A和文件B的inode号码虽然不一样，但是文件A的内容是文件B的路径。读取文件A时，系统会自动将访问者导向文件B。因此，无论打开哪一个文件，最终读取的都是文件B。这时，文件A就称为文件B的"软链接"（soft link）或者"符号链接（symbolic link）。

这意味着，文件A依赖于文件B而存在，如果删除了文件B，打开文件A就会报错："No such file or directory"。这是软链接与硬链接最大的不同：文件A指向文件B的文件名，而不是文件B的inode号码，文件B的inode"链接数"不会因此发生变化。

*ln -s命令可以创建软链接：ln -s 源文文件或目录 目标文件或目录*

4）用户通过文件名，打开文件。实际上，系统内部这个过程分成三步：首先，系统找到这个文件名对应的inode号码；其次，通过inode号码，获取inode信息；最后，根据inode信息，找到文件数据所在的block，读出数据。

使用ls -i命令，可以看到文件名对应的inode号码：ls -i example.txt

5）移动文件或重命名文件，只是改变文件名，不影响inode号码。

## 事件驱动select和epoll

（题目：http://www.nowcoder.com/test/question/done?tid=4814418&qid=44765#summary）

select 和 epoll效率差异的原因：select采用轮询方式处理连接，epoll是触发式处理连接。

Select:

1. Socket数量限制：该限制可操作的Socket数由FD\_SETSIZE决定，内核默认32\*32=1024.

2. 操作限制：通过遍历FD\_SETSIZE(1024)个Socket来完成调度，不管哪个Socket是活跃的，都遍历一遍。

Epoll：

1. Socket数量无限制：该模式下的Socket对应的fd列表由一个数组来保存，大小不限制（默认4k）。

2. 操作无限制：基于内核提供的反射模式，有活跃Socket时，内核访问该Socket的callback，不需要遍历轮询。

但当所有的Socket都活跃的时候，所有的callback都被唤醒，会导致资源的竞争。既然都是要处理所有的Socket，那么遍历是最简单最有效的实现方式。

# 新技术

## 集群

集群是一组相互独立的、通过高速网络互联的计算机，它们构成了一个组，并以单一系统的模式加以管理。一个客户与集群相互作用时，集群像是一个独立的服务器。集群配置是用于提高可用性和可缩放性。

## Hadoop系统分布式存储和并行计算架构

NameNode—JobTracker；DateNode—TaskTracker

## hadoop和spark的区别

缺。

Hadoop除了提供为大家所共识的HDFS分布式数据存储功能之外，还提供了叫做MapReduce的数据处理功能。

Spark是一个基于内存计算的开源的集群计算系统。

## Docker

是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。

# 数学

## 点乘，叉乘

略

## 常用数学公式

见考研时整理的数学笔记

## 特征值的定义

存在一个非零向量X和实数λ，使得AX=λX成立，则称λ为矩阵A的特征值，X称为A相对于λ的特征向量。

## 大数定律，中心极限定理

大数定律：随着试验次数的增多，事件发生的频率逐渐稳定于该事件的概率。

中心极限定理：大量微小的随机变量之和的极限分布渐近于正态分布

## 全概公式，贝叶斯公式

（略）

## 最大似然估计（MLE，maximum likelihood estimation）

离散型： ，求最大的L。

## 常见分布

离散型：

二项分布：N件商品中有M件次品，进行n次有放回抽样，抽得k件次品概率

几何分布

超几何分布：N件商品中有M件次品，抽取n件，抽得k件次品概率

泊松分布

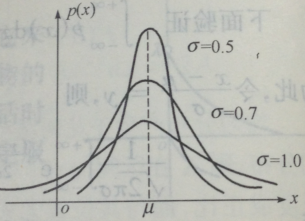
连续性：

指数分布

泊松分布是单位时间内独立事件发生次数的概率分布，指数分布是独立事件的时间间隔的概率分布。（http://www.ruanyifeng.com/blog/2015/06/poisson-distribution.html）

## 正态分布

公式：

 从公式中很容易看出越小，中间部分面积越大。

题目：根据统计，北京市初婚年龄服从正态分布，其均值为25岁，标准差为5岁，问25岁到30岁之间结婚的人；其百分数为多少？（0.3413）

先根据，其中σ即为标准差，然后查表得到为0.8413，最后可求得答案。

附加： ，

## 抽样分布

卡方分布（χ2分布）、t分布和F分布（《概率论与数理统计》p201）

## 最大值最小值概率

最大：

最小：

# 智力题

一些题目：http://www.verydemo.com/demo\_c180\_i24179.html

## 两个鸡蛋从高楼落下问题

递减数列

## 四个装药丸的罐子问题

略

## 5只小白鼠喝蒸馏水，一瓶有毒，测试液体瓶数

Cn1+Cn2+ Cn3+…+ Cnn-1+Cnn＝2n- Cn0＝2n- 1.

## 9个球中一个次品用天平称3次找出次品球

略，网上答案都不靠谱，自己想想就行。

若已知次品球重，只用秤两次。

## 12个球中一个次品用天平称3次找出次品球

略，较为复杂

## 证明:如果p为质数且p>3,则数p^2-1可被24整除

首先,p>3 为质数,因此 p 是奇数,设 p=2n+1 ,

则 p^2-1=(2n+1)^2-1=4n^2+4n=4n(n+1) ,

由于 n、n+1 是连续正整数,其中一个必为偶数,

因此 p^2-1 能被 8 整除；

其次,p>3 为质数,则 p 被 3 除余 1 或 -1（也就是 p 不能被 3 整除） ,设 p=3m±1 ,

则 p^2-1=(3m±1)^2-1=9m^2±6m=3(3m^2±2m) 能被 3 整除,

所以,p^2-1 能被 8\*3=24 整除 .

## 25匹马，5个跑道，最少比多少次能比出前3名？

最快跑7场.前5场决出各组名次.第6场将每组的第一名放在一起比赛.此时的第一名即为所有马中的第一名,无需再参加之后的比赛.假设第6场中前三名的马分别来自x,y,z三个组.第7场由x2,x3,y1,y2,z1比赛,决出两名,即为所有马中的第二,第三名.

## 平面内有11个点，由它们连成48条不同的直，由这些点可连成多少个三角形

平面中有11个点，如果这些点中任意三点都没有共线的，那么一共应该有C(11，2)=55。经分析可知有一个三点共线，一个四点共线。

那么，这11个点能组成的三角形的个数为，C(11，3)-C(3，3)-C(4，3)=165-1-4=160

## 50个红球和50个黑球，有两个一模一样的桶，往桶里放球，随机抽，采用什么策略可以抽到红球的概率更高？

一个罐子放一个红球,另一个罐子放49个红球和50个蓝球。

## 有100个球，一次最多拿5个，最少拿一个，A先拿B后拿， 谁能拿到最后一个就赢

第一个拿了后第二个拿肯定可以让2人共拿6个，所以A先拿4个，然后让B开始拿，自己跟着拿（6-B拿的个数）

## 有一个随机数产生函数能产生5个随机数，怎么产生1-7的随机整数

利用rand5()函数生成1-25之间的数字，然后将其中的1-21映射成1-7，丢弃22-25。例如生成(1，1)，(1，2)，(1，3)，则看成rand7()中的1，如果出现剩下的4种，则丢弃重新生成。

## 10000!末尾有多少个零?

2499。

http://zhidao.baidu.com/link?url=SO0t6TeQ2weM1XfOJC0sJEt7y5zqm4VA-rrzkn3SsGQApKkW-Hp7wa6pdMacsvxa8VlS0ZAycyUa6IM\_ySnbP\_

## 有两路公交车，如果只等A车,平均需要5分钟；如果只等B车,平均需要7分钟。假定两辆车运行时间独立，那么平均需要等多长时间才能等到车

35分钟内一共来了12辆车，平均每 35/12 min 来一辆

## 从10双不同的鞋子中任取8只，若取出的鞋子中没有成对的，那么有多少种不同取法？

先从10双里取出8双，有45种方法。

再8双每双两只任取1只有2\*2\*2\*2\*2\*2\*2\*2=256

最后将45\*256=11520

## 一幅扑克牌，摸到黑牌就继续摸下去，直到摸到红牌为止，平均能摸到几张黑牌

https://www.zhihu.com/question/19823409

如果牌张数是无限多的，那么就等于1；如果牌张是有限多的，那么会略小于1

## 行测

练习：http://exam.chinagwy.org/?mod=xingce&act=train

1. 数字推理

若数字值变化较大，当前数1）与前两数之积有关2）与平方数有关

# 其他

## 图灵奖获得者

1983 肯·汤普逊，丹尼斯·里奇 UNⅨ操作系统和C语言

1998 詹姆斯·尼古拉·格雷（失踪） 数据库与事务处理

2000 姚期智 计算机理论方面：伪随机数生成理论、密码学

## 常浏览的技术类网站

CSDN，博客园cnblogs，伯乐在线，我爱机器学习(52ml.net)，我爱计算机(52cs.org)，微博-视觉机器人，github，微信公众号

## 编译原理

编译程序结构，每部分含义 《南师编译原理》 p2-3

语法分析：自顶向下，自底向上

## 加密算法

（ 《南师计算机网络 p366》 ）

1）对称加密：加密与解密使用的是同样的密钥，所以速度快，但由于需要将密钥在网络传输，所以安全性不高。常见的对称加密算法有DES、3DES、Blowfish、IDEA、RC4、RC5、RC6和AES。

2）非对称加密：使用了一对密钥，公钥与私钥，所以安全性高，但加密与解密速度慢。常见的非对称加密算法有RSA、ECC（移动设备用）、Diffie-Hellman、El Gamal、DSA（数字签名用）。

RSA密钥一般推荐使用1024位。

3）Hash算法：对目标信息生成一段特定长度的唯一的Hash值，却不能通过这个Hash值重新获得目标信息。因此常用在不可还原的密码存储、信息完整性校验等。常见的Hash算法有MD2、MD4、MD5（Message-Digest Algorithm 5，信息-摘要算法5）、HAVAL、SHA。

（华为2016 X Summer第一场的解密题采用了base64，第二场的解密题采用了AES。小知识：AES加密后的字符串前缀一般都是U2FsdGVkX…，用base64解密后是Salted…。）

## 正则表达式

略

## 几个哈希

感知哈希算法（perceptual hash algorithm）

一致性哈希算法（consistent hashing）

局部敏感哈希（licality-sensitive hashing）

## javascript变量

没有用var，其实是定义了一个全局变量

## 你知道的开源协议有哪些？

（http://blog.csdn.net/u014680729/article/details/24382261）

BSD（Berkeley Software Distribution，伯克利软件分发许可协议）：给于使用者很大自由的协议。可以自由的使用，修改源代码，也可以将修改后的代码作为开源或者专有软件再发布。

GPL（GNU General Public License，GNU通用公共许可协议）：GPL许可证是自由软件的应用最广泛的软件许可证，人们可以修改程式的一个或几个副本或程式的任何部分，以此形成基於这些程式的衍生作品。必须在修改过的档案中附有明显的说明：您修改了此一档案及任何修改的日期。 您必须让您发布或出版的作品，包括本程式的全部或一部分，或内含本程式的全部或部分所衍生的作品，允许第三方在此许可证条款下使用，并且不得因为此项授权行为而收费。

BSD、GPL和Unix的关系见《STL原码剖析》p7.

## 你知道的开源软件有哪些？

JDK，Eclipse，MySQL

Linux，Hadoop

## 拼写纠错

检测

1. 构造字典树，判断单词是否存在
2. 使用hash，判断单词是否存在

纠错

1. LCS
2. 对于中文纠错，根据拼音
3. 机器学习角度考虑

## 优化网站加载速度

http://www.iteye.com/news/30202

以及CSS spirit

## PV、UV、IP之间的区别与联系

http://playkid.blog.163.com/blog/static/56287260201361951919690/

PV（page view）即页面浏览量

UV（unique visitor）即独立访客数

IP可以理解为独立IP的访问用户

## 随机数的产生

一般计算机的随机数都是伪随机数，以一个真随机数（种子）作为初始条件，然后用一定的算法不停迭代产生随机数。一般种子可以是当前的系统时间。

## 内存，磁盘，cache等访问速度

L1 cache reference 读取CPU的一级缓存 0.5 ns

L2 cache reference 读取CPU的二级缓存 7 ns

Main memory reference 读取内存数据 100 ns

## PE（Portable Executable）

意为可移植的可执行的文件，常见的EXE、DLL、OCX、SYS、COM都是PE文件，PE文件是微软Windows操作系统上的程序文件（可能是间接被执行，如DLL）

# 英语

## 专业英语

A、logarithmic 对数的

B、linear 线性度

C、quadratic 二次的

D、exponential 指数的

----------------------------------

recursive 递归的

polymorphic 多态的

bipartite graph 二分图

----------------------------------

parameter 形参

argument 实参

## 自我介绍

I'm Jinyang Liu, a third-year post-graduate student from Nanjing University, majoring in computer science and technology. I like programming and often use C plus plus. In addition, I interest in data structure and algorithms. Last two years, I did research on computer version and machine learning. I did some projects related to face recognition.

I'm Liu Jinyang,a junior student from Nanjing Normal University,majoring in computer science and technology.

I like programming,especially good at C plus plus.In the past few years,I study java,C#,SQL.I learned web programming by myself and I can use html,css and php.

项目

职务

OK,that’s all.Thank you!

## 优点，缺点

My advantage is that I like to explore. When I have problem, I like to think.

My disadvantage is easy to forget things。To solve this problem, I am used to make notes.

## 职业规划

（无）

## 介绍你喜欢的课程；假设对方现在是一位大一新生，讲授这门课

I like C++ most.

Today, I will teach you the C++. C++ is one of the most popular programming languages and is implemented on a wide variety of hardware and operating system platforms. /\*It has three characters: Encapsulation, Inheritance and Polymorphism.\*/

If you want to study well, you should exercise frequently.

## 喜欢的运动

My favorite sport is football. When I was a child, I often play football with my friends. Football gives me much sweet memory.

## 休闲时最喜欢做什么

When I am leisure, I like watching movie. I like American movie, especially those directed by Quentin Tarantino. His movie is splendid.

## 描述本科的大学生活

My university life is not only the period for us to learn the professional knowledge, but also the period for us to develop our comprehensive abilities. The campus life is really different from what I have experienced in high school. Campus life is rich and colorful. Firstly, I have more free time to learn what I want to do. I have to learn to be independent, including study and life. I have to study by myself, nobody will supervise me. Secondly, there are many activities that we can take part in training us to learn to how to communicate well with others around us.

## 感兴趣的研究方向？why？

I have broad interests. For example, image processing and artificial intelligence.

I can use this technology to resolve many problems and they are interesting.

## 介绍下做过的项目，毕业设计

I have done Student Score Management System. It’s used to management student’s score. Both teachers and students can use it.

Graduation project: Transaction processing system.

# 投行

## 解释“期货（future）”

（暂略）

# 读书

《程序员面试宝典》

《王道程序员求职宝典》 内容简单，适合用来过一遍题目

《编程之美》

《剑指offer》 （面试微软这本书很重要，要会白纸上写代码）

《Effective C++》

《Cracking the Coding Interview》（中文版为《程序员面试金典》）

# 博客

[程序员面试、算法研究、编程艺术、红黑树、数据挖掘5大系列集](http://blog.csdn.net/v_july_v/article/details/6543438)

[求职面试 - 程序媛想事儿- 博客频道 - CSDN.NET](http://blog.csdn.net/lanxuezaipiao/article/category/1879745)

# 试题搜集

**题目**

[牛客网2016春招备战学习计划](http://www.nowcoder.com/plan/2016-springintern?jobId=640)

[大数据技术Hadoop面试题，看看你能答对多少？](http://dataunion.org/2256.html)

[sql语句练习50题](http://blog.sina.com.cn/s/blog_790bb7190101biql.html)

[115个Java面试题和答案——终极列表（上）](http://www.importnew.com/10980.html)

[115个Java面试题和答案——终极列表（下）](http://www.importnew.com/11028.html)

自己的个人博客，CSDN博客，知乎收藏，浏览器收藏夹，微博，github-star，Notebook笔记，纸质笔记

**公司真题：**牛客网http://www.nowcoder.com/contestRoom

**面试题：**http://www.nowcoder.com/discuss/4614?from=wb

# 面试

**面试准备**

准备纸笔 （打印一份简历，随时查看）

投的什么岗位

电话面试：手机充好电 （准备耳机）

**面试问题**

1. 自我介绍
2. 项目
3. 描述项目（基于什么平台，用了哪些技术，实现了哪些算法，对项目组的贡献）
4. 项目中用到的算法
5. 项目遇到的问题，解决方法
6. 项目难点
7. 有什么想问问题

**提问题**

我的表现

公司工作环境，上班时间，到岗时间，工作内容

面试官所在组，所做内容

**HR面**

工作地点，是否接受调剂

想去的部门，未来职业规划

平时喜欢做其他那些事情

你觉得你最挫败的事是什么？最有成就的是什么？

生活或学习中有没有什么事让你特别有激情？

同学眼里你是怎样的，用三个词来描述？你自己认为呢？也用三个词。（乐观，自省，顽强；内向不善言辞，太过追求完美，太过随和显得软弱）

你觉得你最大的特点是什么？怎么证明？

最近读的书？《浪潮之巅》《怪诞行为学》《我的遥远的清平湾-史铁生》《经济学原理 微观经济学分册-曼昆》

和其他团队合作，但对方的人不配合怎么办？（压力面）

安排团队出游？要有调理，不要想到一点讲一点，考虑到细节更好。（搜狗技术面问题）

# 个人经验

注重和面试官的交流，代码题要先理清思路再开始写。

发内推邮件时，邮件一定要写正文。

面试过程可能有压力面，要积极与面试官交流。

讲话放慢语速！！！

# 求职网站

学校论坛：[小百合](http://bbs.nju.edu.cn/)，[上交饮水思源](https://bbs.sjtu.edu.cn/frame2.html)，[复旦日月光华](http://bbs.fudan.edu.cn/)，[北大未名](http://cs.bdwm.net/)，[浙大飘渺水云间](http://proxy3.zju88.net/agent/board.do?name=Work&mode=0&page=0)，[水木社区](http://www.newsmth.net/nForum/#!board/Career_Campus)

学校就业创业信息网：[南大就业创业信息网](http://job.nju.edu.cn/)，[上交就业信息网](http://scc.sjtu.edu.cn/)，[北大就业指导中心](http://scc.pku.edu.cn/)，

其他：[牛客网](http://www.nowcoder.com/)，[拉勾网](http://www.lagou.com/)，

# 北美求职

在线协同编辑：<http://collabedit.com/> （百度实习生电话面试采用了这个）

面试经验：

1. WANG YUANXUAN.复旦：<http://blog.yxwang.me/2012/12/job-hunting-in-usa-1/>

2.刘佳倩.清华<http://blog.sina.com.cn/s/blog_72f8a5d00102v2z7.html>

# 他人经验

马孔多.北大： <https://www.douban.com/note/465541650/>

江少华. 蚂蚁金服：

<http://mp.weixin.qq.com/s?__biz=MzAwNjM1ODkxNQ==&mid=401890161&idx=1&sn=da6795bae6c28a4af99cae4d7f6c2339&scene=0#rd>

大菜B. 黑龙江大学：<http://www.nowcoder.com/discuss/5908>

chkkch.复旦. Facebook：<http://www.cnblogs.com/chkkch/p/3401015.html>

yijun li.中大.google：<https://www.zhihu.com/question/48635051/answer/117345462>

2014 IT校招待遇大揭秘：<http://www.voidcn.com/blog/marelin/article/p-3387503.html>

# 招聘要求

携程-机器学习/数据挖掘

2年或者以上机器学习/人工智能研究或实践经验

在以下至少一个领域有深入的研究：

a)统计/机器学习（回归(Logistic)，聚类，时间序列，分类树，Boosting，SVM，Random Forest，神经网络等）；推荐算法（Collaborative Filtering, SVD++）；

b)文本挖掘和语义分析(Na?ve Bayes, LDA, 知识图谱表示，结构化预测，语义解析，信息检索，知识挖掘等）

熟悉和掌握算法模型训练软件之一(如R, SAS, SPSS,或者matlab等）。

熟练下列数据库查询之一Hive、SQL Server、Oracel、MySQL、或者Teradata等。

良好的逻辑思维能力，能够从海量数据中发现有价值的规律。

七牛-大数据开发工程师：

本科及以上学历，计算机相关专业，基础扎实

熟悉 Linux 的 Shell 命令，灵活运用 Shell 做文本处理和系统操作

熟练掌握一门或多门编程语言，参与过大型项目者优先，熟悉 Golang，Scala 者优先

具有丰富的海量数据 ETL 加工处理经验

有从事分布式数据存储与计算平台应用开发经验，熟悉 Hadoop，Spark，HBase，Cassandra 及 Hive 等

有从事实时计算的经验，如 Spark Sreaming 及 Storm 等

给著名开源项目提交过代码者优先，如 Hadoop，Spark 及 Docker 等

优秀的沟通与表达能力及较强的自我驱动能力

# <说明>

这个文档创建于大四准备考研复试的时候，研究生期间进行了大量补充。以面试题为主，也有少部分题目主要出现在笔试中，如图像处理和linux部分，在准备面试时这些部分可以不看。

因为自己擅长的是C++，所以其他语言涉及的较少。