Objective-C葵花宝典第一重(内功篇)--类与对象

> 切记:欲练神功，挥刀自宫；炼丹服药，内外齐通。
>
> 今练气之道，不外存想导引，渺渺太虚，天地分清浊而生人，人之练气，不外练虚灵而涤荡昏浊，气者命之主，形者体之用。天地可逆转，人亦有男女互化之道，此中之道，切切不可轻传。修炼此功，当先养心，令心不起杂念，超然于物外方可，若心存杂念，不但无功，反而有性命之忧。

概述

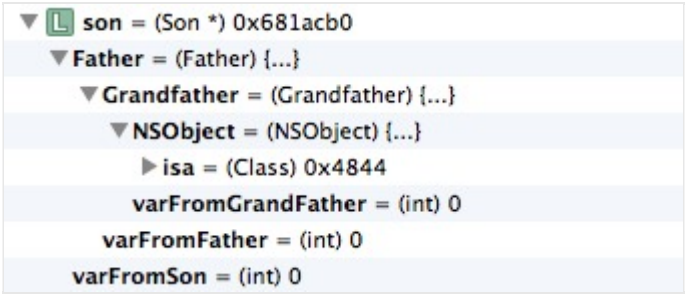
对象

Objective-C中,类和对象的底层数据结构,可以参考Objective-C底层数据结构.

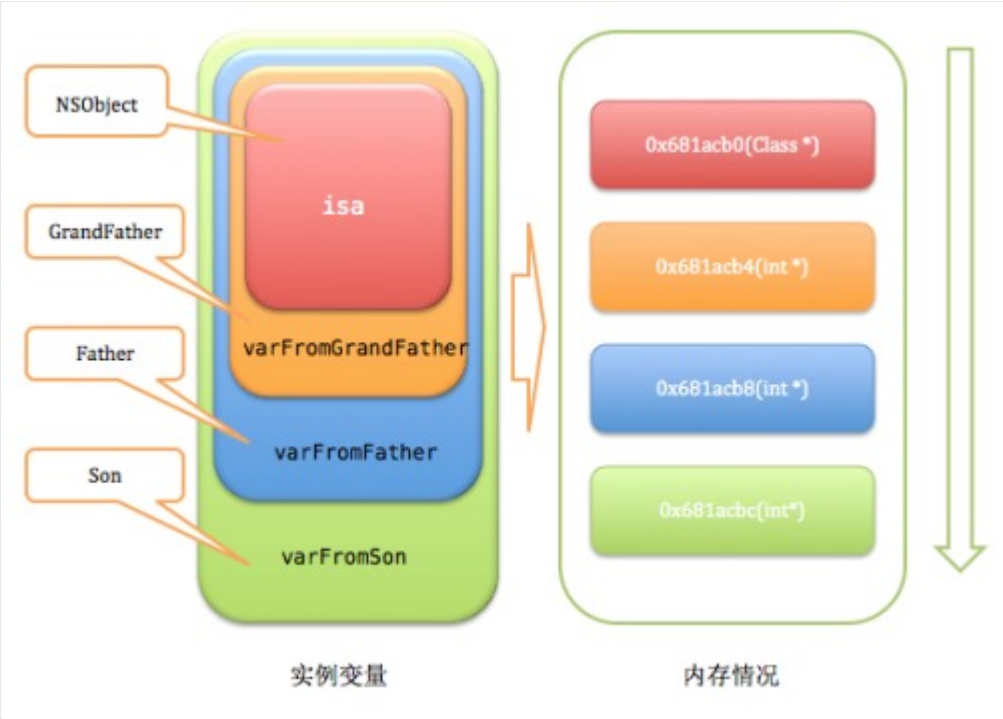
其中对象是通过struct objc_object结构实现

```
typedef struct objc_object *id;
```

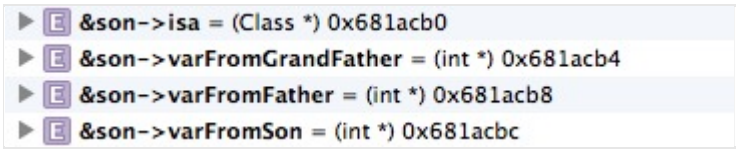
对象在运行时候,会产生如下图的结构形式



我们把这个转换成种更容易理解的数据组织形式



真正的内存形式是这样的



从对象内存形式上来看.对象内的变量成员,是从祖类继承而来(子成父业啊),在对象内部生成副本.从对象的内存组织来看,对象本身并不关心行为(对象的方法或实例方法),重点都在数据的组织上.

```
    (_)   ______     _____ _
   /  /    /  ___/    /  __  `/
  /  /    (__    )    /  /_/  /
/_/    /_____/     \__,_/
```

对象的都有一个isa实例变量,它是从继承层次最高的NSObject继承而来.isa是表示对象的关键.在Objective-C中,是不是第一等对象,isa就是其标志,就好像<变形金刚>中的汽车人都有一个

如果你发现的铁家伙是这个标志

那你就赶紧逃命去吧~

isa是一个指针,指向了该对象的类.实质上,同一个类的实例,都指向同一个类对象(类也是一种特殊对象).类中包含了实例方法,也就是说,同一个类的所有实例共用了这些实例方法.消息就是发送给对象,对象转交给其isa指向类去处理.这种现象类似于当下火热的云计算.

云鸡一算,要风得风,要雨得雨

Objective-C的这种设计,既可以友好地实现面向对象,又可以有效地节约内存.降低冗余数据.对象对方法的调用是通过isa间接去调用,这样就造成了方法调用的动态性,主要原因是:

1. 一个对象并不晓得它能否应答一个方法,它本身既不包含方法的实现,也不包含有方法的指针,而是间接通过isa转到自己的类才能知道
2. 类中的实例方法是以链表形式存在,运行时候,可以修改链表中的实例方法,即可以增删改查,这与C中的函数默认都是extern的不同

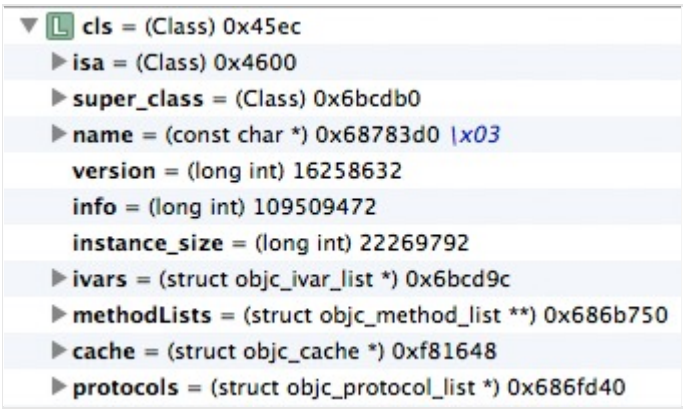
从上面的分析,isa的意义就更加重大,倘若没有isa,一个对象就跟char,int类型等没啥区别了,不具备回应消息的本领.所以说,isa是什么,就好比是古时候官员的乌纱帽,有乌纱帽,就有权力,乌纱帽没了,就是凡夫俗子

类

类是通过struct objc_class结构实现的,

```
typedef struct objc_class *Class;
```

在运行时,将会产生如下图这样的结构



在Objective-C的世界里,一看到isa的第一个反应就是,咦,对象.是的,Objective-C中的类其实也是一种对象.

> 天地不仁，以万物为刍狗<道德经>

既然天地生的万物,那么天地又是何物?

如果人类是上古神仙女娲所造,那女娲又因何而生呢?

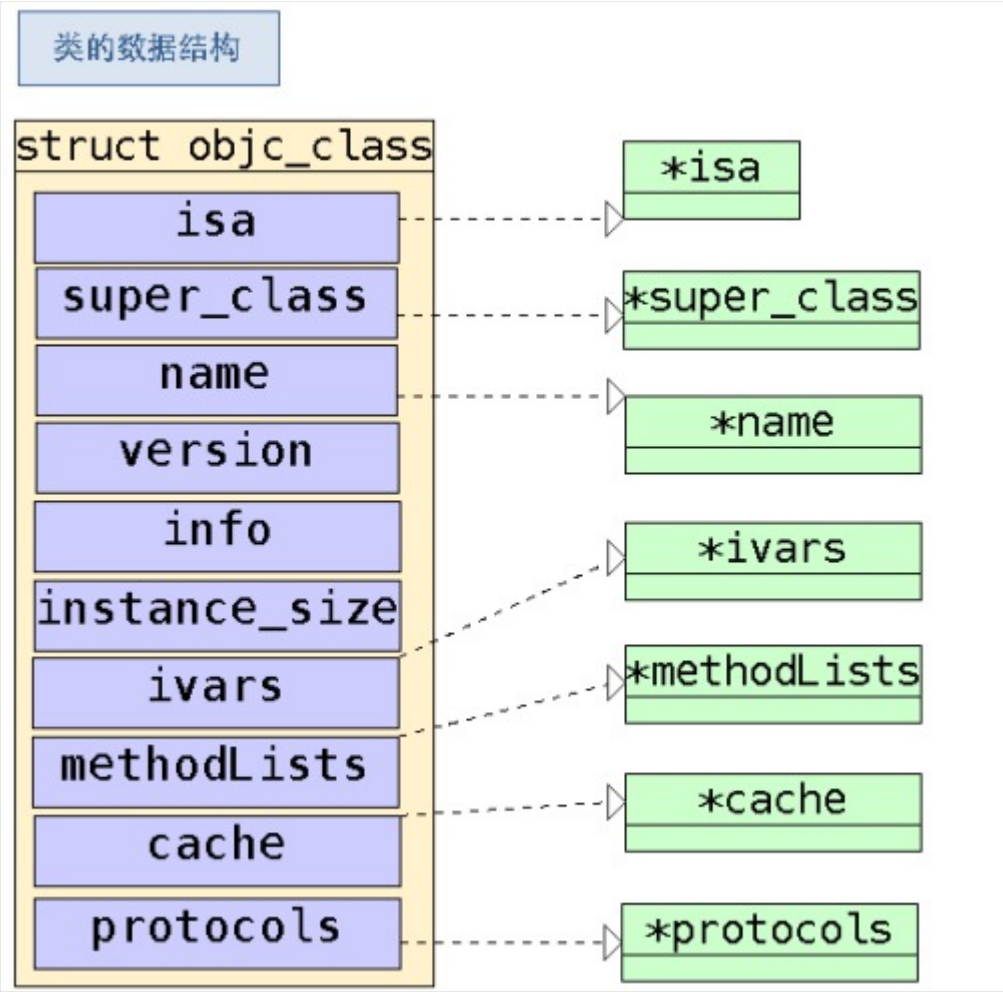
一般程序语言,对象的尽头都是自己生了自己.自己下个蛋,爬出来了自己.似乎是悖逆的,但确实如此.作为根类的NSObject就是这样一个家伙!

这看起来确实很困惑,但是譬如生物学中的造血干细胞可谓细胞的制造厂,但是造血干细胞又是谁制造的呢,咦,也是造血干细胞

首先,类对象也是一种对象,那么它也会有自己的行为,这种行为称作类方法.与一般的类实例一样,类对象也不具备处理类方法的能力,也是要借助isa找到它所属的类,既元类,去调用类方法,类对象本身也是专注于数据的存储和布局形式.可参考类和元类

但与常规的类实例不同,一般而言类对象的字段是固定的.即它默认情况下总是包含isa,super_class,name,version,info,instance_size,ivars,methodLists,cache,protocols.

一个类的数据形式会是如下



而它的数据在内存中的组织形式也是线性的

我们知道,在Objective-C中,我们一般的定义形式是声明一个类的实例变量,属性,实例方法和类方法.并不能声明类变量.所以一般而言类对象的数据形式就是如上图的那些固定的字段.

除了我们介绍的isa,剩余的字段的含义如下:

- super_class 指向父类的指针.因为Objective-C借鉴了SimTALK,在类的继承实

现上,是通过一条继承链实现的.super_class就是整个继承链的核心字段.

- name 类的名字
- version 版本
- info 信息
- instance_size 实例的内存大小
- ivars 是一个指向实例变量列表的指针
- methodLists 是一个指向实例方法列表的指针
- cache 缓存了常用的实例方法
- protocols 是一个指向协议列表的指针

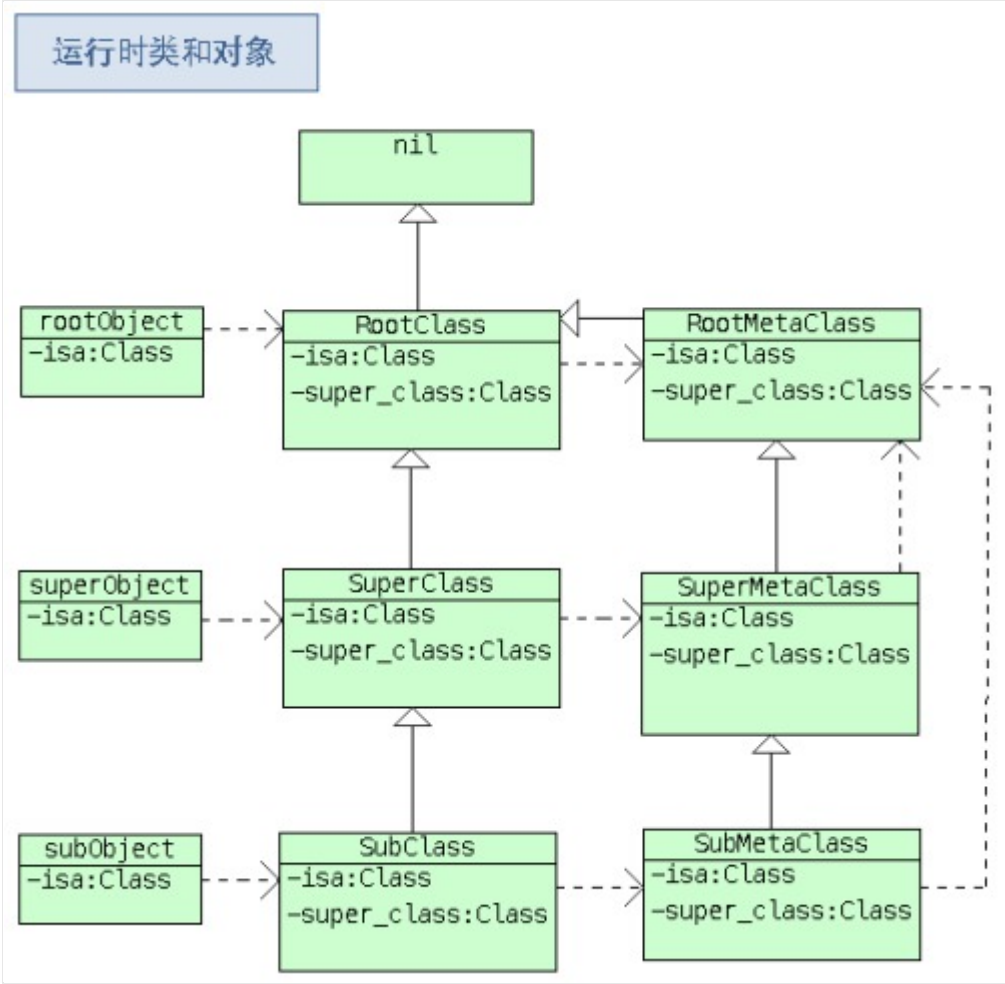
我们可以这样简单地去解释运行时的Objective-C的数据结构的含义

1. 对象与实例变量有关,对象自身存储着实例变量
2. 类对象与实例方法有关,实例方法必须通过类对象才能知晓
3. 元类与类方法有关, 类方法必须通过元类才能知晓

对象,类与元类可谓与道教里的三清一般哦~~

运行时的类和对象

运行时的类和对象如下图



一般而言

- isa关于对象是什么类
- super_class关于继承链
- 所有元类都有同样的元类,因为他们的isa都指向同一个根元类

参考

- Objective-C底层数据结构
- 类和元类