Android-RefBase&sp&wp

作者：wind
原文链接：https://ld246.com/article/1441015871125
来源网站：链滴

最近在读android源码，学到不少有用的东西，写篇博文记录一下。
首先介绍一下RefBase，sp，wp：

    RefBase是所有对象的鼻祖类似于Java中的Object，它结合了sp和wp，实现了一套通过引用计数方法来控制对象生命周期的机制。那么sp和wp分别是什么呢？网上有很多资料把他们解释为smart pointer，也就是智能指针，其实并不准确。其实sp就是strong pointer，wp则是weak pointer，分别做强指针和弱指针。下面通过源码来具体介绍一下：

<pre class="prettyprint lang-java">RefBase::RefBase():mRefs(new weakref_impl(this)){}</pre>

    这是RefBase的构造函数，它给成员变量mRefs赋值为weakref_impl类型对象，暂且把它叫做影子象。

<pre class="prettyprint lang-java">weakref_impl(RefBase* base):mStrong(INITIAL_STRONG_ALUE)//强引用计数，初始值0x1000000
                        ,mWeak(0) //弱引用计数，初始值0
                        ,mBase(base) //
                        ,mFlags(0)
                        ,mStringRefs(NULL)
                    {}</pre>

    可以看到一个RefBase对象中包含了一个影子对象，影子对象里包含了强弱引用计数。下面看sp对:

<pre class="prettyprint lang-java">sp<T>::sp(T* other):m_ptr(other)
{
  if(other) other->incStrong(this);
}</pre>
<pre class="prettyprint lang-java">sp对象需要用RefBase对象来初始化，所以other就是RefBase所代表的对象。接着看RefBase类中的 incStrong函数。</pre>

<pre class="prettyprint lang-java">void RefBase::incStrong(const void* id) const
{
  weakref_impl* const refs=mRefs;
  refs->addWeakRef(id);
  refs->incWeak(id);
  ......
  refs->addStrongRef(id);
  const int32_t c=android_atomic_inc(&refs->mStrong);//原子+1操作，返回旧值，即参
+1，返回+1前的值
  if(c!=INITIAL_STRONG_VALUE){
    return;
  }
  android_atomic_add(-INITIAL_STRONG_VALUE,&refs->mStrong);//原子加，参数相加
  const_cast<RefBase*>(this)->onFirstRef();
}</pre>

意图很明显，sp对象初始化后，影子对象的弱引用计数0+1=1，强引用计数变为1。

    看完sp的构造之后，来看一下wp的构造

<pre class="prettyprint lang-java">wp<T>::wp(const sp<T>& other):m_ptr(ot
er.m_ptr)
{
  if(m_ptr){
    m_refs=m_ptr->createWeak(this);
  }
}</pre>

wp的用sp对象初始化，所以other是sp的对象，m_ptr指向的是实际对象。createWeak之后，跟刚
一样，弱引用计数+1，所以现在mWeak=2 mStrong=1。

    下面看一下wp的析构函数

<pre class="prettyprint lang-java">wp<T>::~wp()
{
  if(m_ptr) m_refs->decWeak(this);
}</pre>

调用的是RefBase对象的decWeak()函数。主要作用就是弱引用计数-1，即从2变成1。

  接着sp的析构：

<pre class="prettyprint lang-java"> if(m_ptr) m_ptr->decStrong(this);</pre>

这里把强引用计数和弱引用计数都变成0了。由于是否delete是根据强弱引用计数来判断的，所以实对象和影子对象都被delete掉。强引用计数为0，实际对象被delete，弱引用计数为0，影子对象被delete。

这就是比较基本的RefBase、sp、wp的关系啦，其实对象的生命周期，还有其他的因素，不过都是围这三个对象和强弱引用计数来进行的。

