关于 ThreadLocalMap 的理解

作者：catty
原文链接：https://ld246.com/article/1506518117279
来源网站：链滴

在多线程编程中，有时会用到ThreadLocal类来处理一些变量。这个类的作用是为每个线程创建一个量的副本，以达到变量线程安全的目的，所以这个类名叫ThreadLocal，可以翻译为线程局部变量。

查看ThreadLocal的源码，会发现其中有个名为ThreadLocalMap的内部类，其实现原理类似hashMa，那么我们很自然的会想到ThreadLocalMap应该是把每个线程Thread作为key，而变量作为value储起来的。

但是当看到Threadlocal中set方法的时候，估计大部分人会感到疑惑，为什么是这样的？set方法源码下：

```
public void set(T value) {
    Thread t = Thread.currentThread();
    ThreadLocalMap map = getMap(t);
    if (map != null)
        map.set(this, value);
    else
        createMap(t, value);
}
```

map.set(this, value)，为什么是this，而不是当前线程t呢？按照常规想法，map以当前线程为key，量为value储存，取的时候也以当前线程为key取，不是刚好吗？

据说早期JDK确实是这么设计的，后来改成这样是因为性能原因。如果使用Threadlocal维护一个ThredLocalMap，为每个线程创建一个变量副本，那么当线程很多时，ThreadLocalMap也会变得很大，验告诉我们，一般用同一个数据结构存储大量数据的时候，效率都会变得很低，当然，事实也确实是样。

后来的设计就是上面的代码了，查看ThreadLocalMap map = getMap(t)中的getMap()方法会发现返回值是Thread里的一个ThreadLocalMap属性，也就是说，并不是由ThreadLocal去维护一个大的TreadLocalMap，而是由每个Thread自己去维护一个小的ThreadLocalMap。为什么说是小的ThreadocalMap？因为Thread里的ThreadLocalMap是以ThreadLocal为键，变量为值，也就是说map里只一个元素，通过把一个大map拆成若干个小map来提升效率，并且这个变量既然是线程局部变量，那由Thread来维护，从关系上来看也很清晰。