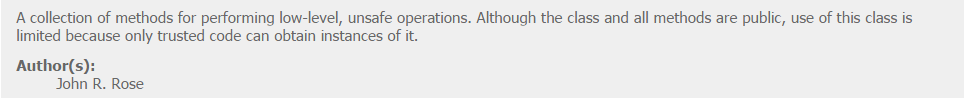
Unsafe



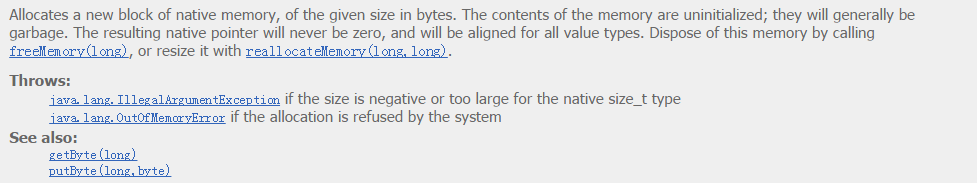
JDK/sun.misc.Unsafe

提供了硬件级别的原子操作，非堵塞原子方式来减少线程调度开销，

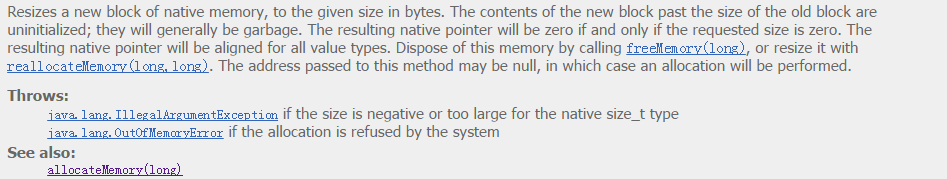
Java中不直接使用Unsafe.java，封装成原子类AtomicInteger给用户使用，但是可以反射获得。

1.内存管理：

public native long allocateMemory(long bytes);



public native long reallocateMemory(long address, long bytes);

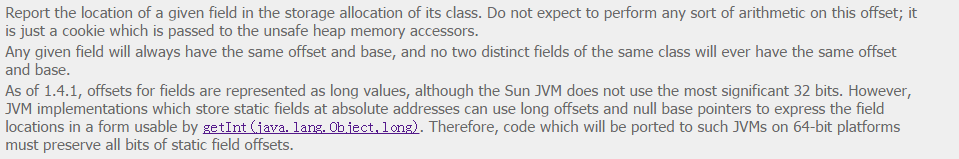


public native void freeMemory(long address);

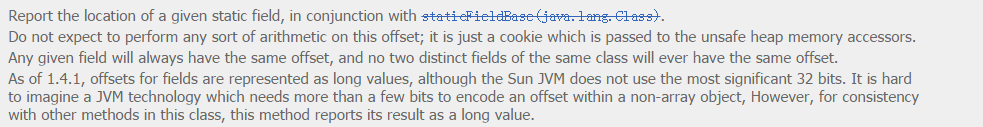


2.定位对象某字段的内存位置，也可以修改对象的字段值，即使它是私有的

public native long staticFieldOffset(Field f);

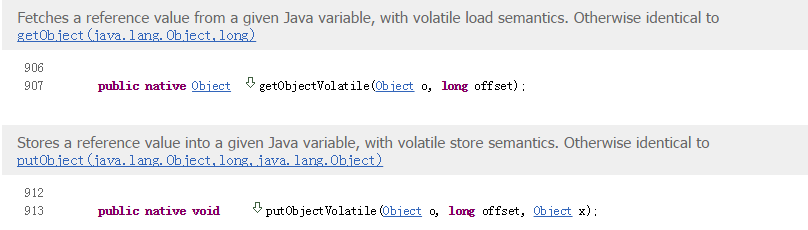


public native long objectFieldOffset(Field f);



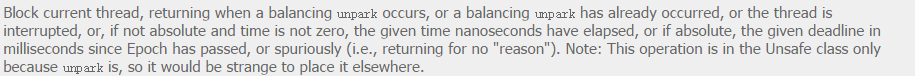
public final native boolean compareAndSwapObject(Object o, long offset,Object expected, Object x);



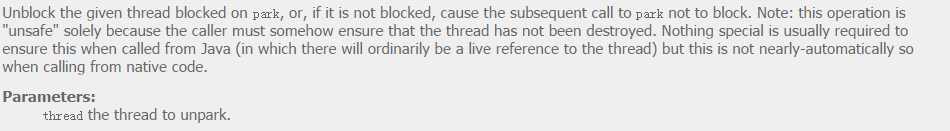


3.恢复和挂起

public native void park(boolean isAbsolute, long time);



public native void unpark(Object thread);



将一个线程进行挂起是通过park方法实现的，调用 park后，线程将一直阻塞直到超时或者中断等条件出现。unpark可以终止一个挂起的线程，使其恢复正常。整个并发框架中对线程的挂起操作被封装在 LockSupport类中，LockSupport类中有各种版本pack方法，但最终都调用了Unsafe.park()方法。

在以前的jdk版本中是通过wait,notify方法实现的。该方法也是通过底层在某个信号量上的阻塞队列实现的。而Unsafe类中直接提供操作系统调度命令park,unpark,减少信号量的开销

chronicle项目:

mmap机制来实现高效的读写数据，号称每秒写入5到20百万条数据。

作者有个测试，写入1百万条log用时0.234秒，用java自带的logger，用时7.347秒。

<https://github.com/peter-lawrey/Java-Chronicle>

利用Unsafe来直接读写内存，从而提高效率

<https://github.com/peter-lawrey/Java-Chronicle/blob/master/chronicle/src/main/java/com/higherfrequencytrading/chronicle/impl/UnsafeExcerpt.java>

参考：

<http://blog.csdn.net/aesop_wubo/article/details/7537278>

<http://blog.csdn.net/hengyunabc/article/details/7657934>

CAS = compareAndSwap