MICROOH 麦可网

Android-从程序员到架构师之路

出品人：Sundy
讲师：高焕堂（台湾）

http://www.microoh.com

C05_d

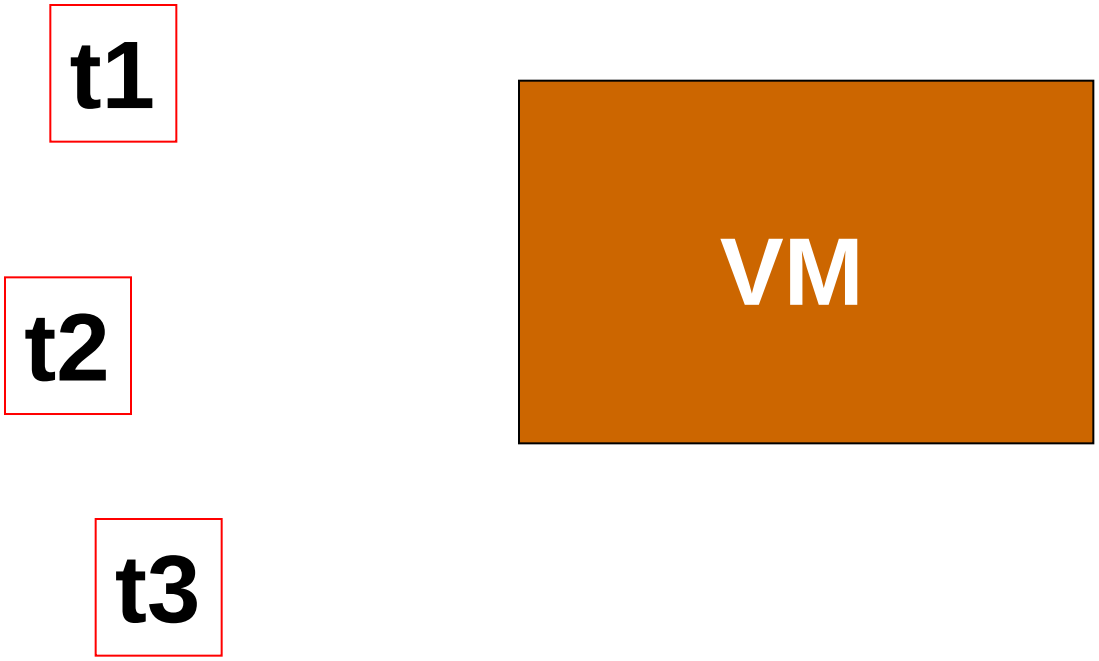
JNI：多个Java线程进入本地函数(d)

By 高煥堂

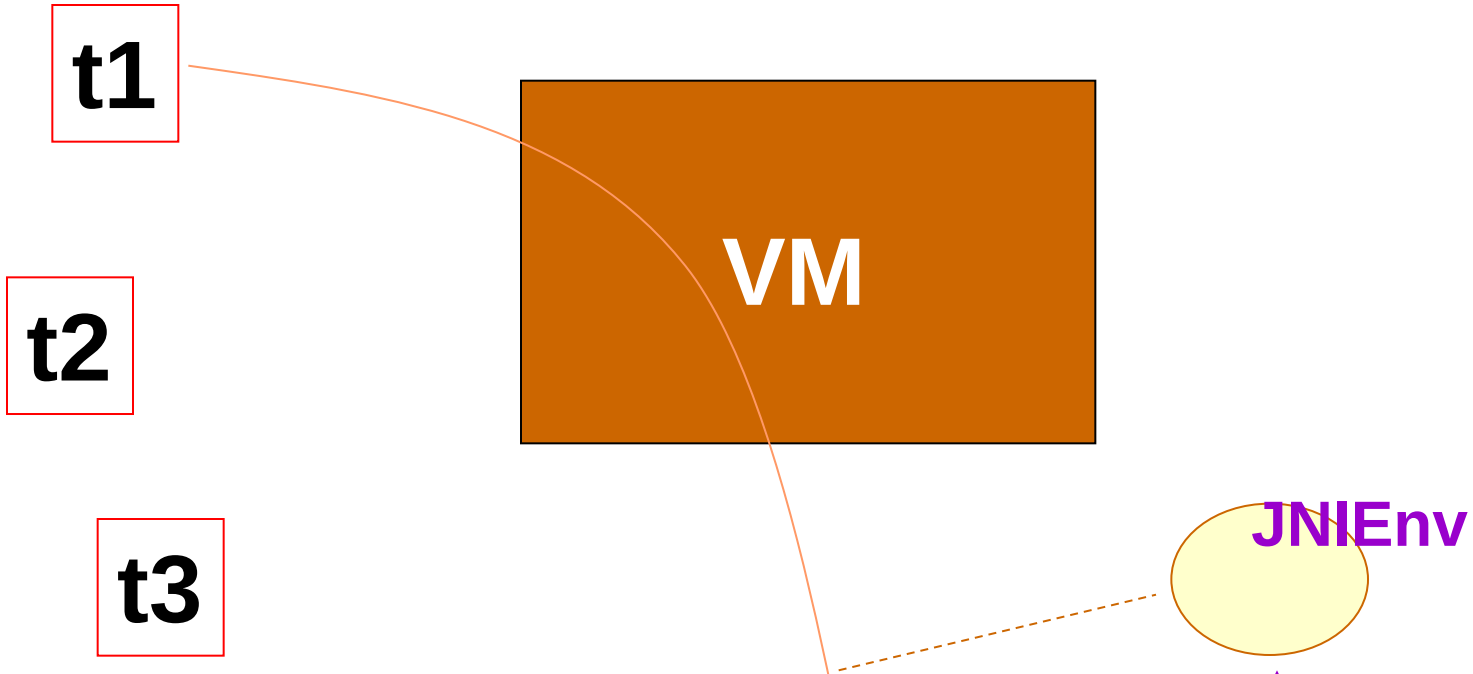
4、本地函数的线程安全

多个线程同步(Synchronization)

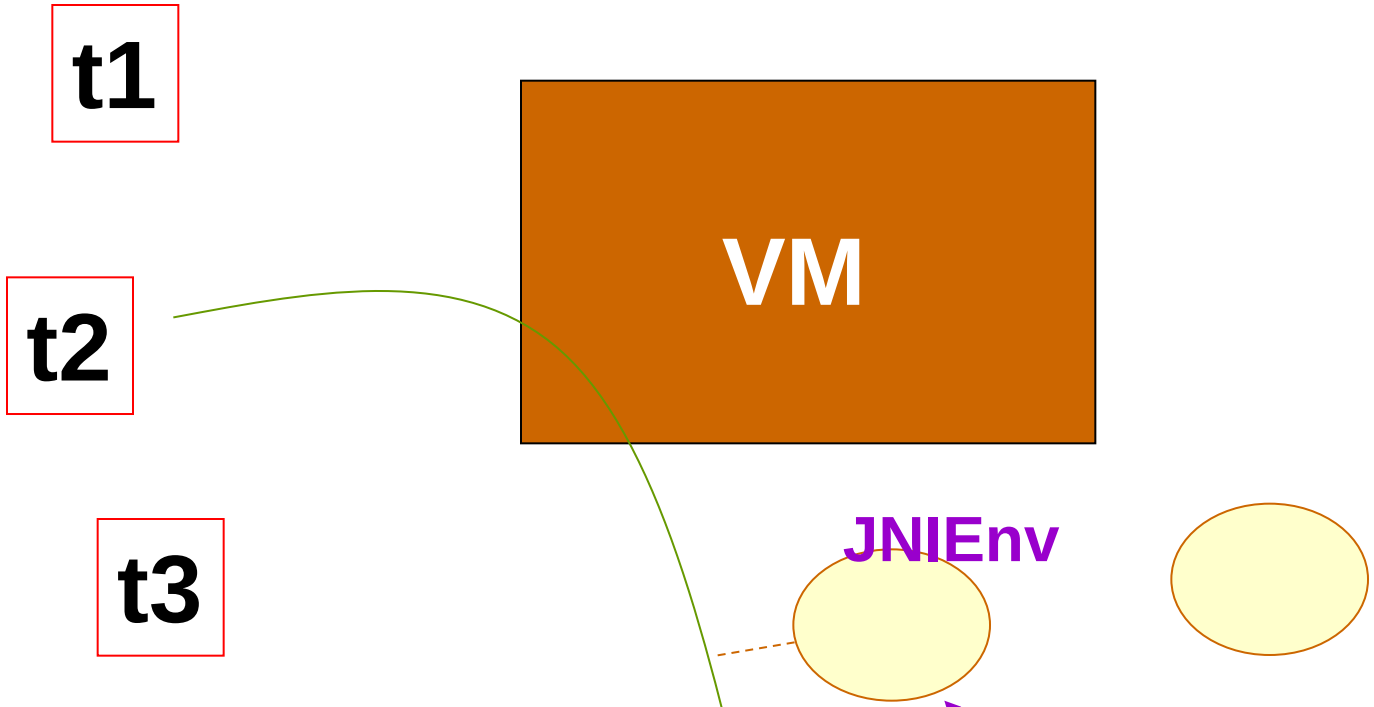
前言




```
int sum;
JNIEXPORT void JNICALL
Java_com_misoo_thread_JTX03_execute(JNIEnv *env,
        jobject thiz){
   sum = 0;
   for(int i = 0; i<=10; i++)  sum += i;
}
```




```
int sum;
JNIEXPORT void JNICALL
Java_com_misoo_thread_JTX03_execute(JNIEnv *env,
        jobject thiz){
   sum = 0;
   for(int i = 0; i<=10; i++)  sum += i;
}
```




```
int sum;
JNIEXPORT void JNICALL
Java_com_misoo_thread_JTX03_execute(JNIEnv *env,
        jobject thiz){
   sum = 0;
   for(int i = 0; i<=10; i++)  sum += i;
}
```

- Java程序可能会有多个线程几乎同时先后进入同一个本地函数里执行。
- VM会替各线程创建其专用的JNIEnv对象，有些平台允许你将私有的数据储存于JNIEnv的对象里，避免共享问题；但有些平台则否。

- 如果你的私有数据不能或不想將它存于JNIEnv对象里，而是放在一般的变量里，就必须自己注意变量共享而产生的线程安全问题了。

```
/*  com_misoo_thread_JTX03.cpp    */
// .........
int sum;
JNIEXPORT jstring JNICALL
Java_com_misoo_thread_JTX03_execute(JNIEnv *env, jobject thiz){
   sum = 0;
   for(int i = 0; i<=10; i++){
          sum += i;
          Thread_sleep(1);
         }
    env->CallStaticVoidMethod(mClass, mid, sum, 0);
    sprintf(sTid, "%lu", 0);
    jstring ret = env->NewStringUTF(sTid);
    return ret;
}
```

- 當多个线程几乎同时先后进入此本地函数execute()里执行，由于sum等变量是公用的，就可能发生线程安全问题了。

当会发生线程冲突时，又如何呢?

化解冲突的范例

- 解决途径之一是：多个Java线程之同步(Synchronization)

两个线程(并行)执行**execute()**函数

```
// JTX04.java
// .........
public class JTX04 {
  .........
public long calculate(){
            Thread t1 = new Thread(){
                public void run() {
                    JTX04.this.execute(JTX04.this);
                }
            };
            t1.start();
            try { Thread.sleep(2000);
            } catch (InterruptedException e) {
                    e.printStackTrace();
          }
```

```
        String ss = execute(this);
        ac01.ref.setTitle("ss: " + ss);
        return 0;
    }
    .....
    private native void Init(Object weak_this);
    private native String execute( Object oSync );
}
```

看谁先抢到这个对象的钥匙(Key)

```
/*  com_misoo_thread_JTX04.cpp    */
// ........
JavaVM *gJavaVM;
jmethodID   mid;
jclass      mClass;     // Reference to JTX04 class
jobject     mObject;    // Weak ref to JTX04 Java object to call on
char sTid[20];
unsigned int e1;
int x;
int sum;
long test;
```

```
JNIEXPORT jstring JNICALL
Java_com_misoo_thread_JTX04_execute(JNIEnv *env, jobject thiz,
                                    jobject syncObj){
   env->MonitorEnter( syncObj );
   sum = 0;
   for(int i = 0; i<=10; i++)  {
           sum += i;   Thread_sleep(1);
          }
   env->CallStaticVoidMethod(mClass, mid, sum, 666);
   env->MonitorExit( syncObj );
   long pid = getpid();
   sprintf(sTid, "%lu", test);
   jstring ret = env->NewStringUTF(sTid);
   return ret;
}
// .........
}
```

- 执行到指令：

  JTX04.this.execute(JTX04.this);

  和

  String ss = execute(this);

- 都把目前的Java 对象(即JTX04对象)传递给本地的execute()函数。

- 先进入execute()的线程先执行到指令：

  env->MonitorEnter(syncObj);

- 也就向JTX04对象索取钥匙(Key)。由于JTX04对象只要一把钥匙，所以其它后进入的线程只好停下来等待。

- 当执行到指令：

  env->MonitorExit(syncObj);

- 也就把钥匙(Key)交还给JTX04对象，让等待中的其它线程可以逐一进入。

Thanks...

高煥堂