aardio 范例: 线程共享变量

```
//线程共享变量
import console.int;
import thread.table;

//线程都独享全局变量环境，下面创建一个多线程共享变量。
var thrdVar = thread.var();
thread.invokeAndWait(
    function(thrdVar){

        //修改线程共享变量的值
        thrdVar.set(123456);

    },thrdVar//线程变量可以传入其他线程使用
)

//获取线程共享变量的值
console.log(thrdVar.get());

//及时释放不再使用的线程共享变量
thrdVar.release();

//thread.table 则是创建了一个线程间共享的 table 对象，
var thrdTable = thread.table()

var thrdHandle1 = thread.create(
    function(thrdTable){

    thrdTable.push( "线程ID:" + thread.getId() )

    //用下标读写线程共享表的键值，键名与对象的属性、函数名同名不会冲突。
    thrdTable["b"] = 34;
    thrdTable["c"]= 56;

    //可以读写普通表，但子表并不是线程共享表。
    thrdTable["d"] = {};

    //每次读取 thrdTable.d 都会得到新的副本
    var localData = thrdTable["d"];

    //修改 localData 不会自动会同步到线程共享资源。
    localData.tab = { name = "name" }

    //下面这样只是修改子表的临时副本，子表不是线程共享表
    thrdTable.d.tab = { name = "name" }

    //只有修改线程共享表的直接成员，才会自动同步到线程共享资源
    thrdTable["d"] = localData;

    //注意 # 操作符不适合用于线程共享表
    var len = #thrdTable

    //应当用 len 函数或 length 属性获取共享数组长度
    len = thrdTable.len(); // 也可以用 thrdTable.

    //线程共享表可作为参数传入其他线程
    },thrdTable
)

//等待线程执行完,并关闭所有线程句柄
thread.waitClose(thrdHandle1)

//读取线程表的成员
console.log( thrdTable.pop(2) )

//读取线程表的成员
console.log( thrdTable.a );

//遍历共享表
for( k,v in thrdTable.each() ){
    console.dump(k,v);
}

//不使用的线程共享表应及时释放。
thrdTable.release();
```

Markdown 格式