# aardio 范例: 多线程 —— 入门

```
//入门
import win.ui;
/*DSG{{*/
var winform = win.form(text="多线程 —— 入门";right=536;bottom=325)
winform.add(
button={cls="button";text="启动线程";left=27;top=243;right=279;bottom=305;db=1;dl=1;dr=1;font=LOGFONT(h=-16);z=1};
edit={cls="edit";left=27;top=20;right=503;bottom=223;db=1;dl=1;dr=1;dt=1;edge=1;multiline=1;z=2}
)
/*}}*/

winform.button.oncommand = function(id,event){

    //禁用按钮并显示动画
    winform.button.disabledText = {"✶";"✷";"✸";"✺";"✸";"✶"}

    //创建工作线程
    thread.invoke(

        //线程启动函数
        function(winform){

            for(i=1;3;1){
                sleep(1000); //在界面线程执行 sleep 会卡住

                //调用界面控件的成员函数 - 会转发到界面线程执行
                winform.edit.print("工作线程正在执行，时间: " + tostring( time() ) );
            }

            winform.button.disabledText = null;

        },winform //窗口对象可作为参数传入工作线程
    )
}

/*
建议阅读《aardio 多线程开发入门》
https://bbs.aardio.com/forum.php?mod=viewthread&tid=13625

一、做好心理准备
虽然 aardio 的多线程开发非常简单，但是:
1、请先了解:「多线程」开发比「单线程」开发更复杂这个残酷的现实。
2、请先了解: aardio 这样的动态语言可以实现真多线程非常罕见。

建议先找任意的编程语言试试能不能更轻松地实现 aardio 多线程范例相同的效果。
如果实践后你发现 aardio 做多线程开发要轻松得多，那么请继续往下看，
如果遇到一点困难和限制就 …… 什么啥的，建议早点放弃。

二、简单了解什么是线程
当你点击EXE文件系统一个应用程序的时候 - 系统会创建一个进程（process）
而在一个进程内可以包含多个线程(thread)。用来显示界面的线程，我们通常称为"界面线程"，
其他不是用来显示界面的线程，我们一般称为"工作线程"或者是"后台线程"。
进程的启动线程称为「主线程」，「界面线程」通常是主线程。

每个线程可以按单一顺序执行一系列的任务 —— 但不能并发执行多个任务。
多个线程就可以并发执行多个任务。

三、为什么需要多线程
界面线程会使用 win.loopMessage() 启动一个消息循环，
win.loopMessage() 就象一个快递公司不知疲倦地处理分发、处理窗口消息，
直到最后一个非模态、非 MessageOnly 的独立窗口（ 或 mainForm ）关闭后才会退出消息循环。
当然你也可以使用 win.quitMessage() 退出消息循环。

界面线程如果遇到耗时操作，就会停下来无法继续分发处理消息 —— 这时候界面就表现为「卡死」状态。
这时候如果创建工作线程去执行耗时操作，就可以让界面线程继续分发处理消息（不会卡住）。

四、线程同步的风险

多线程就象多个在并列的轨道上疾驰的火车，如果 A 火车与 B 火车上的人想操作同一部手机（ 线程共享变量 ），
你不能直接从 A 火车上把手伸出去跟B 火车上的人拉拉扯扯。这时候只能先让所有的火车都停下来，互动完了再继续往前开。
需要互动的时候先停下来 —— 这在多线程开发里就是线程同步锁机制。
在 aardio 中用 thread.lock() 创建同步锁，但实际上在 aardio 中很少需要用到同步锁。

上面这种看起来省事的方式会制造大量的麻烦。更好的方法是 A 火车、B 火车上的人都玩自己的手机，而不是共享一部手机。
大家拿着自己的手机（ 线程独享变量 ）相互联系，一样可以愉快地互动。

这就是 aardio 多线程的基本规则：每个线程有独立的运行上下文、独立的全局变量环境，
```

一个线程中的对象传到另一个线程 —— 只会传值而不会传址（传引用）。

注意：
1、aardio 也提供操作共享变量的 thread.get,thread.set,thread.var,thread.table 等。
2、一个对象传入另一个线程虽然是传值，但仍然可能引用同一个可以在线程间安全共享的资源，例如 thread.command,thread.event 等。
3、窗口或控件对象从一个线程传到另一个线程实际上是传同一个窗口句柄的引用对象 —— 在多线程中操作同一个窗口或控件
总是会转发到创建窗口的界面线程内执行。

四、aardio 多线程开发基本规则

多线程开发时要谨记以下基本规则。

1、非主线程的错误信息默认只会输出到控制台，
只有用 console.open() 或 io.open() 打开控制台才能看到非主线程的错误信息。

2、每个线程有独立的运行上下文、独立的全局变量环境，有独立的堆栈。
一个线程不会使用另一个线程的全局变量，一个线程也不会使用另一个线程引入的库。

3、不是所有对象都可以从一个线程传到另一个线程使用。
没有任何外部依赖的数值、字符串、buffer、table、function 可以传入其他线程使用。
这些对象在传入另一个线程时通常会复制值 - 也就是传值而非传址（传引用）。

「类」不可以从一个线程传入另一个线程使用。
「类」创建的实例对象，除非文档有特别说明一般不可以传入另一个线程使用。

win.form 创建的窗体对象以及该窗体上创建的控件对象都可以作为参数传入其他线程。
在其他线程调用窗体与控件对象的成员函数时 —— 都会回发到创建窗体的界面线程执行。
利用这种奇妙的特性 —— 实际上可以在工作线程调用界面线程的任意代码。

COM 对象不可以从一个线程传递到另一个线程。

以下对象可从一个线程传递到另一个线程：
time,time.ole,thread.var,thread.table,
thread.command,thread.event,thread.semaphore,process.mutex,
fsys.file,fsys.stream,fsys.mmap,raw.struct …… 请参考相关文档说明。

如果不想看文档，直接判断一个对象是不是可以跨线程传递的方法也很简单：
不支持传入线程使用的对象，那么传入线程后调用必然会失败报错。
*/

```
winform.show();
win.loopMessage();
```

Markdown 格式