aardio 范例: 多线程界面回调

```
//多线程界面回调

/*
win.ui 创建的窗口（或控件）对象都可以传入工作线程使用。
在工作线程中可调用窗口成员函数，读写窗口属性。
所有调用都会发回界面线程执行（多线程发指令，单线程干活，没有线程交互与同步的负担）。

界面线程中的非窗口对象也可以用 thread.callable 开启此能力。
thread.callable 会创建一个不可见的 MessageOnly 窗口用于代理转发调用消息。
也可以直接在对象的元表中指定 _serialize = ..thread._callableSerialize 以开启此功能。

窗口对象（winform ）被传到线程中以后实际上只是一个影子对象，
如果把他的值打印出来，实际上它只有一个 hwnd 的成员字段指明了窗口句柄，

线程中 winform  的作用就如同一个搞外包的。
你每次在线程中调用他的成员函数，他就会调用 thread.callWnd 呼叫界面线程中那个真正的窗体对象：
“喂！喂！请停下来，外包任务你接不接........”

界面线程中的窗体收到这条消息以后很高兴：“只要给钱（参数），当然可以接的.......”
界面窗体在接收到调用消息以后，使用 thread.applyCallWnd(this,wParam,lParam) 将消息解析为函数调用。
*/

import console;
import win.ui;

//用win.form创建一个线程外包对象
var ☎ = win.form();
☎.messageOnly();//告诉他不需要显示窗口，只保留能处理消息的功能
☎.count = 0;

//添加一个线程外包函数
☎.printf = function(...){
    ☎.count = ☎.count + 1;
    if(☎.count >= 100 ) win.quitMessage()

    console.printf(...)
}

//循环执行
for(i=1;10;1){

    //创建线程
    thread.invoke(
        function(☎){

            var tid = thread.getId();

            for(i=1;10;1){

                //调用外包对象的成员函数，让他外包给界面线程执行
                ☎.printf("线程ID:%d 循环次数:%s",tid,i);
                sleep(10);
            }

        },☎/*扔到线程参数里*/
    )
}

win.loopMessage();

console.log("这里用一个控制台演示并不需要窗体，同样可以实现线程间的外包调用");
console.log("但是实际开发中，应在窗体线程中使用，控制台并不适合用来实现消息循环")
console.pause();
```

Markdown 格式