链滴

检测 Lua 脚本中的死循环

作者：localvar
原文链接：https://ld246.com/article/1522147783147
来源网站：链滴

论坛上有人问，所以把以前做的东西拿出来秀一下。

Lua是一门小巧精致的语言，特别适用于嵌入其它的程序为它们提供脚本支持。不过脚本通常是用户写的，很有可能出现死循环，虽说这是用户的问题，但却会造成我们的宿主程序死掉。所以检测用户本中的死循环并中止这段脚本的运行就显得非常重要了。

可是，一个现实的问题是死循环并不好检测，一些隐藏较深的死循环连人都很难找出来，更不用说让器去找了。所以实际采用的方案多是检测脚本的执行时间，如果超过一定的限度，就认为里面有死循，我下面的例子也是用的这种方法。

以下是几个相关的全局变量（我是喜欢把C++当C用的程序员，C++的忠实粉丝请忍耐一下 mile）的定义。

```
lua_State* g_lua = NULL;          // lua脚本引擎
volatile unsigned g_begin = 0;       // 脚本开始执行的时间
volatile long g_counter = 0;       // 脚本执行计数, 用于判断执行超时
volatile long g_check = 0;         // 进行超时检查时的执行计数
```

run_user_script用来执行用户脚本，它首先通过GetTickCount把当前的时间记录到g_begin中去。后将g_counter加一，在执行完用户脚本后再将其加一，这样就可以保证执行用户脚本时它是个奇数而不执行时是偶数，检测脚本超时的代码可以籍此来判断当前是否在执行用户脚本。还要注意调用用脚本要使用lua_pcall而不是lua_call，因为我们中止脚本的执行会产生一个Lua中的“错误”，在C/C+中它是一个异常，只有用lua_pcall才能保证这个错误被Lua脚本引擎正确处理。

```
int run_user_script( int nargs, int nresults, int errfunc )
{
   g_begin = GetTickCount();
   _InterlockedIncrement( &g_counter );
   int err = lua_pcall( g_lua, nargs, nresults, errfunc );
   _InterlockedIncrement( &g_counter );
   return err;
}
```

下面的check_script_timeout用来检测脚本超时，需要在另外一个线程中周期性的调用，原因我想就用解释了吧。它首先检查是否在执行用户脚本，或者是否已经让当前执行的用户脚本中止过。然后看段脚本执行了多长时间，超过限度就把当前脚本计数记录到g_check中去，并通过lua_sethook设置个钩子函数timeout_break，这个钩子函数会在用户脚本执行时被调用。

```
void check_script_timeout()
{
   long counter = g_counter;

   // 没有执行用户脚本, 不检查超时
   if( (counter & 0x00000001) == 0 )
      return;

   // 已经让当前执行的用户脚本中止了
   if( g_check == counter )
      return;

   // 如果执行时间超过了设置的超时时间(这里是1秒), 终止它
   if( GetTickCount() - g_begin > 1000 )
   {
      g_check = counter;
```

```
        int mask = LUA_MASKCALL | LUA_MASKRET | LUA_MASKLINE | LUA_MASKCOUNT;
        lua_sethook( g_lua, timeout_break, mask, 1);
    }
}
```

最后就是那个钩子函数了，它首先把钩子去掉，因为这个钩子只要执行一次就行了。由于设置钩子和行钩子是在不同的线程中，并且钩子从设置到执行需要一定的时间，所以它要通过对比g_check和g_counter来判断是否还在运行判断超时所执行的那段脚本，不是就什么也不做，是就通过luaL_error产生个错误，并中止脚本的执行，而这个错误最终会被run_user_script中的lua_pcall捕获。

```
void timeout_break( lua_State* L, lua_Debug* ar )
{
    lua_sethook( L, NULL, 0, 0 );
    // 钩子从设置到执行, 需要一段时间, 所以要检测是否仍在执行那个超时的脚本
    if( g_check == g_counter )
        luaL_error( L, "script timeout." );
}
```

上面的检测使用了两个线程，其实在一个线程中也可以做到，并且更简单。但那样会导致钩子函数频执行，影响效率，如果对性能没什么要求的话，也可以采用。