GameGuardian官方文档

GameGuardian 手机修改器 改内存

脚本文档

前言

- 翻译者: 石乐志
- 翻译了十二个小时, 主要是排版有点废时间, 再加上英文水平太差,如有错漏,可以自行改正,如果实在受不了, 有种,你顺着网线来打我呀.

概论

- 欢迎来到GameGuardian脚本文档,你可以在GameGuardian中编写使用多种多样的破解脚本,本文档描述和介绍了能在GameGuardian脚本文件中出现和使用的函数和类.你也可以在我们的论坛: http://gameguardian.net/forum/topic/17447-lua-scripting/中参与讨论.

我们应该怎么样开始学习?

- 我们通过"边学边做"的方式开始编写脚本,往往效果很好, 因为这样我们学得既快速,又富有乐趣.可以通过我们的站点http://gameguardian.net/forum/files/category/6-lua-scripts/下载脚本,开始修改和学习以作为你们学习和成长的起点.
- 反复试验运行的脚本的效果,将使您更容易理解底层的关键思想和概念,然后根据自己的需要回头阅读文档的相关内容.
- 你们也可能想要阅读Lua文档,并在GameGuardian论坛开始提问.

列出所有API方法

- 你们可以在GG修改器列表中使用下面的代码,列出所有的方法.
  `print(gg)`

更多资源信息

- 用GameGuardian破解的相关视频教程例子,可以见官方文档的video导航条目中.网址:https://gameguardian.net/forum/gallery/category/2-video-tutorials/
- GG修改器脚本是基于lua编程语言,下面是关于Lua编程语言学习的补充.
  - 学习更多lua知识,lua官方网站http://www.lua.org/about.html
  - 已经确认的非常好的Lua文档所在地址 http://www.lua.org/docs.html. 该文档提供了所有lua编程语言的的所有信息,是不可缺少的用于提升我们GG脚本编写水平的资源.
- 脚本例子,可见网址 http://gameguardian.net/forum/files/category/6-lua-scripts/

GG使用的帮助文档

- 待补充

API类:都属于gg类成员

函数接口

alert

- alert()方法原型:

```
int alert(string text,
  string positive='ok',
  string negative=nil,
  string neutral=nil,
)
```

- 函数功能:显示一个包含几个按钮的对话框.
- 函数参数:
  - text:对话框文本提示信息
  - positive:positive按钮的文本信息,显示在对话框的最右边,选择该按钮返回值1.
  - negative:negative按钮的文本信息,显示在对话框的右边第二个位置,并靠近positive按钮,选择该按钮返回值2.
  - neutral:neutral按钮的文本信息,位置在最左边,离上面两个按钮距离较远,选择该按钮返回值3.
- 返回值:对话框取消返回值0,选择positive按钮返回值1,选择negative按钮返回值2,选择neutral按钮返回值3.
- 使用例子:

```
gg.alert("scaipt ended") --只显示一个ok按钮
gg.alert("scaipt ended","Yes") --最右边只显示一个yes按钮
gg.alert('A or B?', 'A', 'B')            //从左至右显示按钮B,显示按钮A.
gg.alert('A or B or C?', 'A', 'B', 'C')  //显示按钮从左至右,C,B,A.
gg.alert('A or C?', 'A', nil, 'C')   //最左边和最右边分别显示按钮C和按钮A. 不适用按钮可以赋值为nil.
```

bytes

- bytes()函数原型:

```
string bytes ( string  text,
 string  encoding = 'UTF-8'
)
```

- 函数功能:获得指定编码文本的文本字节
- 参数:
  - text
  - encoding参数: 可选的值,‘ISO-8859-1’, ‘US-ASCII’, ‘UTF-16’, ‘UTF-16BE’, ‘UTF-16LE’, ‘UTF-8’
- 返回值:返回第一个参数的字符串,以指定编码文本的字节流表.索引0开始,每个索引存储一个字节.如果是16位编码的格式,那么表的第一个字节是字符码值,第二个字节是16位的高八位.
- 例子:

```
print('UTF-8', gg.bytes('example'))
print('UTF-8', gg.bytes('example', 'UTF-8'))
print('UTF-16', gg.bytes('example', 'UTF-16LE'))
```

choice

- choice()函数原型:

```
mixed choice  (  table   items,
 string   selected = nil,
 string   message = nil
)
```

- 函数功能:从列表中显示一个选择对话框.该列表由table类型参数的item变量组成.

- 函数参数:

  - items: 列表类型({‘A’,‘B’,‘C’,“D”})
  - selected:如果没有指定或指定值为nil,那么这列表不会默认选择指定列表项.
  - message: 选择列表框的标题
- 返回值:如果列表框取消选择返回nil,否则返回选择项的索引值.

- 例子:

```
print('1: ', gg.choice({'A', 'B', 'C', 'D'}))
print('2: ', gg.choice({'A', 'B', 'C', 'D'}, 2))
print('3: ', gg.choice({'A', 'B', 'C', 'D'}, 3, 'Select letter:'))
print('4: ', gg.choice({'A', 'B', 'C', 'D'}, nil, 'Select letter:'))
```

clearResults

- 函数原型:

```
clearResults()  //无参数,无返回值
```

- 函数功能:清空搜索到的值的列表.
- 无参数
- 无返回值

copyMemory

- 函数原型

```
mixed copyMemory  (  long   from,
 long   to,
 int   bytes
)
```

- 函数功能:复制内存
- 函数参数:
  - from:需要复制内存的起始地址
  - to : 需要复制内存的结束地址
  - bytes:需要复制内存单元的数量,数字8表示复制8个字节.
- 返回值: 成功返回真,失败返回字符串"error"
- 例子代码

```
print('copyMemory:', gg.copyMemory(0x9000, 0x9010, 3))
//copies 3 bytes 0x9000-0x9002 to 0x9010-0x9012
```

copyText

- 函数原型:

```
copyText  (  string   text,
 bool   fixLocale = true
)
```

- 函数功能: 复制文本到剪贴板
- 函数参数:
  - text: 需要复制的文本
  - fixLocale: 是否禁用固定区域界限分隔符的标志.true为禁用,false为不禁用.
- 无返回值
- 使用例子:

```
gg.copyText('1,234,567.890')        -- Will copy '1 234 567,890'
gg.copyText('1,234,567.890', true)  -- Will copy '1 234 567,890'
gg.copyText('1,234,567.890', false) -- Will copy '1,234,567.890'```
```

dumpMemory

- 函数原型:

```
mixed dumpMemory  (  long   from,
 long   to,
 string   dir
)
```

- 函数功能:复制指定内存单元到文件.
- 函数参数:
  - from:起始地址
  - to: 结束地址
  - dir:想要保存输出文件的目录.
- 函数返回值:成功返回true,失败返回字符串"error"
- 函数例子:

```
print('dumpMemory:', gg.dumpMemory(0x9000, 0x9010, '/sdcard/dump'))
-- dump at least one memory page into the dir '/sdcard/dump'
```

editAll

- 函数原型

```
mixed editAll  (  string   value,
 int   type
)
```

- 函数功能: 编辑搜索的结果值.编辑全部结果值.调用该方法之前,你必须已经通过getResult导入了结果值,只有值的指定类型才将会用于结果值.
- 函数参数:
  - value: 将要编辑数据值的字符串形式.
  - type:符号常量,指明数据类型. 来源于TYPE_*指针.后面会接受类型符号常量.
- 函数返回值:成功返回改变的结果值的项目数,失败返回字符串error
- 例子代码:

```
gg.searchNumber('10', gg.TYPE_DWORD)
gg.getResults(5)
gg.editAll('15', gg.TYPE_DWORD)
-- with float:
gg.searchNumber('10.1', gg.TYPE_FLOAT)
gg.getResults(5)
gg.editAll('15.2', gg.TYPE_FLOAT)
-- with XOR mode
gg.searchNumber('10X4', gg.TYPE_DWORD)
gg.getResults(5)
gg.editAll('15X4', gg.TYPE_DWORD)
```

getFile

- 函数原型

```
string getFile  (   )
```

函数功能:获取当前正在运行的脚本的名字.

函数参数:无

函数返回值:成功返回当前脚本的文件名.eg:'/sdcard/Notes/gg.example.lua'

getLine

- 函数原型:

```
int getLine(  )
```

- 函数功能: 获取将要执行的脚本的当前行的行号.
- 函数参数:无
- 函数返回值:返回将要被执行脚本的当前行号.
- 函数例子:

```
print(gg.getLine()) // 24
```

getLocale

- 函数原型:

```
string getLocale  (   )
```

- 函数功能:获取在GameGuardian中的当前选择的字符串本地化情况
- 函数返回值:返回当前在GameGuardian中当前选择的字符串本地化.返回值en_US, zh_CN, ru, pt_BR, ar, uk

getRanges

- 函数原型

```
int getRanges  (   )
```

- 函数功能:内存区域作为一个REGION_*指针标志位遮罩返回
- 函数参数:无
- 函数返回值: REGION_*指针标志遮罩位.

getRangesList

- 函数原型:

```
getRangesList  (  string   filter = '' )
```

- 函数功能:获取选择进程的内存区域的列表.
- 函数参数:
  - filter:过滤字符串.如果指定该选项,仅仅返回和过滤器相匹配的结果.该选项支持通配符.^表示数据开头,$表示数据结尾.*表示任意数量的任意字符,?表示一个任意字符.
- 函数返回值:返回存储内存区域的一个列表.每个元素时一个列表字段:state,start,end,type,name, internalName.
- 例子:

```
print(gg.getRangesList())
print(gg.getRangesList('libc.so'))
print(gg.getRangesList('lib*.so'))
print(gg.getRangesList('^/data/'))
print(gg.getRangesList('.so$'))
```

getResult

- 函数原型:

```
mixed getResults ( int  maxCount)
```

- 函数功能:导入结果到结果列表,并作为表返回.
- 函数参数:
  - maxCount:导入结果的最大数量
- 函数返回值:成功返回结果列表,失败返回字符串"error",每个元素都是一个包含三个关键字的列表:address (long), value (string with a value), flags (one of the constants TYPE_*).
- 函数例子:

```
gg.searchNumber('10', gg.TYPE_DWORD)
local r = gg.getResults(5)
print('First 5 results: ', r)
print('First result: ', r[1])
print('First result address: ', r[1].address)
print('First result value: ', r[1].value)
print('First result type: ', r[1].flags)
```

getResultCount

- 函数原型:

```
long getResultCount( )
```

- 函数功能:获取找到结果的数量.
- 函数参数无
- 函数返回值:找到结果的数量.
- 函数例子:

```
gg.searchNumber('10', gg.TYPE_DWORD)
print('Found: ', gg.getResultCount())
```

getSpeed

- 函数原型:

```
double getSpeed ()
```

- 函数功能:从加速中获取当前速度.
- 函数参数无
- 函数返回值:返回从加速中获取的当前速度.

getTargetInfo

- 函数原型

```
mixed getTargetInfo ( )
```

- 函数功能:如果可能的话,获取关于选择进程的信息列表.一系列字段可以是不同的.打印可见和可用字段的结果列表.
  - 可能的字段:firstInstallTime, lastUpdateTime, packageName, sharedUserId, sharedUserLabel, versionCode, versionName, activities (name, label), installer, enabledSetting, backupAgentName, className, dataDir, descriptionRes, flags, icon, labelRes, logo, manageSpaceActivityName, name, nativeLibraryDir, packageName, permission, processName, publicSourceDir, sourceDir, targetSdkVersion, taskAffinity, theme, uid, label.谷歌应用会返回每个字段.
- 函数参数无.
- 函数返回值:返回选择进程的信息列表,或返回nil
- 函数例子:

```
-- check for game version
local v = gg.getTargetInfo()
if v.versionCode ~= 291 then
    print('This script only works with game version 291. You have game version
', v.versionCode, ' Please install version 291 and try again.')
    os.exit()  //退出程序
end
```

getTargetPackage

- 函数原型:

```
mixed getTargetPackage ()
```

- 函数功能:获取选择进程的包名字.
- 函数参数:无
- 函数返回值:返回选择进程的名字字符串.或者返回nil. eg:‘com.blayzegames.iosfps’

getValues

- 函数原型:

```
mixed getValues(table  values)
```

- 函数功能:获取列表的项目的值
- 函数参数:
  - values:一个包含很多个列表的列表,该表中的每个表包含adress和标志字段.(其中一个就是TYPE*符号常量)
- 函数返回值:成功返回一个新表,失败返回"error",表的每个元素是一个包含三个关键字的表,address (long), value (string with a value), flags (one of the constants TYPE_*).
- 函数例子:

```
gg.searchNumber('10', gg.TYPE_DWORD)
local r = gg.getResults(5) -- load items
r = gg.getValues(r) -- refresh items values
print('First 5 results: ', r)
print('First result: ', r[1])
print('First result address: ', r[1].address)
print('First result value: ', r[1].value)
print('First result type: ', r[1].flags)
local t = {}
t[1] = {}
t[1].address = 0x18004030 -- some desired address
t[1].flags = gg.TYPE_DWORD
t[2] = {}
t[2].address = 0x18004040 -- another desired address
t[2].flags = gg.TYPE_BYTE
t = gg.getValues(t)
print(t)
```

getValuesRange

- 函数原型:

```
mixed getValuesRange  (  table   values )
```

- 函数功能:获取传递的表参数的值的内存区域.
- 函数参数:
  values:参数是表类型,该表既可以是地址列表,也可以是一个地址字段的列表.
- 函数返回值:返回一个列表,该列表的每个键,来源于参数的表值,将和一个短区域代码相联系,(例如ch),失败返回字符串error
- 函数例子:

```
print('1: ', gg.getValuesRange({0x9000, 0x9010, 0x9020, 0x9030}))

-- table as a list of addresses
gg.searchNumber('10', gg.TYPE_DWORD)
local r = gg.getResults(5)
print('2: ', r, gg.getValuesRange(r))
-- table as a list of tables with the address field
```

gotoAddress

- 函数原型:

```
gotoAddress  (  long   address )
```

- 函数功能:在内存编辑器中跳转到指定地址.
- 函数参数:
  - adress:希望跳转的地址.
- 返回值无

isPackageInstalled

- 函数原型:

```
bool isPackageInstalled  (  string   pkg )
```

- 函数功能:通过安装包名字判断指定应用程序在操作系统上时候安装.

- 函数参数:
  - pkg: 安装包名字
- 函数返回值:如果已安装返回true,否则返回false
- 函数例子:

```
print('Game installed:', gg.isPackageInstalled('com.blayzegames.iosfps'))
```

isProcessPaused

- 函数原型:

```
bool isProcessPaused  (   )
```

- 函数功能:获取指定进程的暂停状态.
- 函数返回值:如果进程已暂停返回true,否则返回false

isVisible

- 函数原型

```
bool isVisible()
```

- 函数功能:检查GameGuardian的UI时候打开
- 函数参数无
- 如果GameGuardian修改器的UI是打开的返回true,否则返回false

loadList

- 函数原型:

```
mixed loadList  (  string   file,
 int   flags = 0
)
```

- 函数功能:从文件中导入保存列表
- 函数参数:
  - file:要导入到列表的文件.
  - flags:系列标志 LOAD_*
- 函数返回值:成功true,失败返回"error"
- 例子:

```
print('loadList:', gg.loadList('/sdcard/Notes/gg.victum.txt'))
print('loadList:', gg.loadList('/sdcard/Notes/gg.victum.txt', 0))
print('loadList:', gg.loadList('/sdcard/Notes/gg.victum.txt', gg.LOAD_APPEND))
print('loadList:', gg.loadList('/sdcard/Notes/gg.victum.txt', gg.LOAD_VALUES_FREEZE))
print('loadList:', gg.loadList('/sdcard/Notes/gg.victum.txt', gg.LOAD_APPEND | gg.LOAD_VALUES))
```

multiChoice

- 函数原型:

```
mixed multiChoice  (  table   items,
 table   selection = {},
 string   message = nil
)
```

- 函数功能:显示多项选择对话框.
- 函数参数:
  - items: 列表类型,需要显示选择项目
  - selection: 列表类型,给每个选择项指定同样的默认选择状态.如果key是未发现的那么元素将是未选择的.
  - message:多项选择框的标题.
- 函数返回值:如果不选择,直接取消选择对话框,会返回nil,否则返回一个表,表里的元素时一个项目关键字和值true.
- 函数例子:

```
print('1: ', gg.multiChoice({'A', 'B', 'C', 'D'}))
-- show list of 4 items without checked items
print('2: ', gg.multiChoice({'A', 'B', 'C', 'D'}, {[2]=true, [4]=true}))

-- show list of 4 items with checked 2 and 4 items
print('3: ', gg.multiChoice({'A', 'B', 'C', 'D'}, {[3]=true}, 'Select letter:')
)
-- show list of 4 items with checked 3 item and message
print('4: ', gg.multiChoice({'A', 'B', 'C', 'D'}, {}, 'Select letter:'))
-- show list of 4 items without checked items and message
```

processKill

- 函数原型

```
bool processKill( )
```

- 函数功能:暴力杀掉选择进程,注意该操作可能使得该进程的数据丢失
- 函数参数无
- 函数返回值:成功true,失败false

processPause

- 函数原型

```
bool processPause()
```

- 函数功能:暂停选择程序进程
- 函数参数无
- 函数返回值:成功true,失败false

processResume

- 函数原型

```
bool processResume ( )
```

- 函数功能:如果程序进程被暂停,该操作恢复程序进程.
- 函数参数无
- 函数返回值:成功true,失败false.

processToggle

- 函数原型:

```
bool processToggle()
```

- 函数功能:切换选择进程的暂停状态.如果进程是暂停的,就恢复进程,否则暂停进程
- 函数参数无
- 函数返回值:成功返回true,失败返回false

prompt

- 函数原型

```
mixed prompt  (  table   prompts,
 table   defaults = {},
 table   types = {}
)
```

- 函数功能:显示数据入口对方框.对于字段的顺序, 提示必须是数字数组。
- 函数参数:
  - prompts:列表类型,存放着指定键和对输入字段的描述.
  - defaults:列表类型,存放着,给每个键提示的默认值
  - types:列表类型,每个键提示的指定数据类型,可用的值:'number', 'text', 'path', 'file', 'setting', 'speed', 'checkbox',从类型依赖于输入字段附近的附加元素的输出
- 函数返回值: 对话框取消,返回nil,否则返回一个表,存放着提示和keys,和从输入字段中获得的值.
- 代码例子:

```
print('prompt 1: ', gg.prompt(
    {'ask any', 'ask num', 'ask text', 'ask path', 'ask file', 'ask set', 'ask speed', 'checked', 'not checked'},
    {[1]='any val', [7]=123, [6]=-0.34, [8]=true},
    {[2]='number', [3]='text', [4]='path', [5]='file', [6]='setting', [7]='speed', [8]='checkbox', [9]='checkbox'}
))
print('prompt 2: ', gg.prompt(
    {'ask any', 'ask num', 'ask text', 'ask path', 'ask file', 'ask set', 'ask speed', 'check'},
    {[1]='any val', [7]=123, [6]=-0.34}
))
print('prompt 3: ', gg.prompt(
    {'ask any', 'ask num', 'ask text', 'ask path', 'ask file', 'ask set', 'ask speed', 'check'}
))
```

removeResults

- 函数原型

```
mixed removeResults  (  table   results )
```

- 函数功能:从列出的发现结果的列表中,移除结果值.
- 函数参数:
  - result:表类型,每个元素也是一个表类型,该表包含了地址,flags字段.(其中之一是符号常量TYPE_*)
- 返回值:成功返回ture,其它返回error.
- 例子代码:

```
gg.searchNumber('10', gg.TYPE_DWORD)
local r = gg.getResults(5)
print('Remove first 5 results: ', gg.removeResults(r))
```

require

- 函数原型

```
require  (  string   version = nil,
 int   build = 0
)
```

- 函数功能:检查GameGuardian的版本号.如果这个版本号低于需要的版本号,那么脚本将结束,并提示更新GameGuardian版本.
- 函数参数
  - version:最低需求版本号.
  - build:运行脚本的最小版本号.
- 函数返回值:无
- 例子代码

```
gg.require('8.31.1')

gg.require('8.31.1', 5645)
gg.require(nil, 5645)
```

saveList

- 函数原型

```
mixed saveList  (  string   file,
 int   flags = 0
)
```

- 函数功能:将保存列表的内容保存到文件.
- 函数参数:
  - file: 将内容将要保存到的文件
  - flags: SAVE_*系列保存标志.
- 函数返回值:成功true,失败"error"
- 例子代码:

```
print('saveList:', gg.saveList('/sdcard/Notes/gg.victum.txt'))
print('saveList:', gg.saveList('/sdcard/Notes/gg.victum.txt', 0))
print('saveList:', gg.saveList('/sdcard/Notes/gg.victum.txt', gg.SAVE_AS_TEXT))
```

searchAddress

- 函数原型

```
mixed searchAddress  (  string   text,
 long   mask = -1,
 int   type = gg.TYPE_AUTO,
 int   sign = gg.SIGN_EQUAL,
 long   memoryFrom = 0,
 long   memoryTo = -1
)
```

- 函数功能:根据指定参数,执行一个地址搜索.如果在结果列表中没有搜索到结果值,将执行一个新的搜索,否则重定义并重新搜索.
- 函数参数:
  - text: 搜索字符串,格式跟在GameGuardian的gui中操作的字符串一样.
  - mask:遮罩掩码默认是-1(0xFFFFFFFFFFFFFFFF)
  - type: 数据类型符号常量TYPE_*.
  - sign:信号,SIGN_EQUAL 或 SIGN_NOT_EQUAL.

    - memoryFrom:搜索的起始内存地址
    - memoryTo: 搜索的内存的结束地址.
- 函数返回值:True或返回字符串"error"
- 函数例子:

```
gg.searchAddress('A20', 0xFFFFFFFF)
gg.searchAddress('B20', 0xFF0, gg.TYPE_DWORD, gg.SIGN_NOT_EQUAL)
gg.searchAddress('0B?0', 0xFFF, gg.TYPE_FLOAT)
gg.searchAddress('??F??', 0xBA0, gg.TYPE_BYTE, gg.SIGN_NOT_EQUAL, 0x9000, 0xA09
000)
```

searchFuzzy

- 函数原型:

```
mixed searchFuzzy  (  string   difference = '0',
 int   sign = gg.SIGN_FUZZY_EQUAL,
 int   type = gg.TYPE_AUTO,
 long   memoryFrom = 0,
 long   memoryTo = -1
)
```

- 函数功能:用指定的参数优化模糊搜索
- 函数参数:
    - old值和new值的差值,默认是"0"
    - sign:信号常量,符号常量之一是SIGN_FUZZY_*.
    - type: 数据类型符号常量.
    - memoryFrom:搜索的内存单元起始地址.
    - memoryTo: 搜索的内存单元的结束地址.
- 返回值:返回True或失败返回"error"
- 例子代码:

```
gg.searchFuzzy()
-- value not changed
gg.searchFuzzy('0', gg.SIGN_FUZZY_NOT_EQUAL)
-- value changed
gg.searchFuzzy('0', gg.SIGN_FUZZY_GREATER)
-- value increased
gg.searchFuzzy('0', gg.SIGN_FUZZY_LESS)
-- value decreased
gg.searchFuzzy('15')
-- value increased by 15
gg.searchFuzzy('-115')
-- value decreased by 115
```

searchNumber()

- 函数原型:

```
mixed searchNumber  (  string   text,
 int   type = gg.TYPE_AUTO,
 bool   encrypted = false,
 int   sign = gg.SIGN_EQUAL,
 long   memoryFrom = 0,
 long   memoryTo = -1
)
```

- 函数功能:根据指定参数,搜索一个数字.在结果列表中如果没有搜索到结果,那么将执行一个新的搜索,否则重新定义搜索.

- 函数参数
  - text: 搜索字符串,该字符串格式跟GameGuardian的GUI格式相同.
  - type: 数类型,符号常量表示TYPE_*
  - sign: 信号,其中一个符号常量SIGN_*
  - memoryFrom: 搜索内存单元的起始地址.
  - memoryTo: 搜索内存单元的结束地址.
- 函数返回值: 成功true,失败返回字符串error
- 代码例子:

```
gg.searchNumber('10', gg.TYPE_DWORD)
-- number search
gg.searchNumber('-10', gg.TYPE_DWORD, true)
-- encrypted search
gg.searchNumber('10~20', gg.TYPE_DWORD, false, gg.SIGN_NOT_EQUAL)
-- range search
gg.searchNumber('6~7;7;1~2;0;0;0;0;6~8::29', gg.TYPE_DWORD)
-- group search with ranges
```

setRanges

- 函数原型:

```
setRanges  (  int   ranges )
```

- 函数功能: 设置内存区域,通过设置REGION_*标志的遮罩(掩码)位.
- 函数参数:
  - ranges: REGION_*标志的遮罩位.

setSpeed

- 函数原型

```
mixed setSpeed  (  double   speed )
```

- 函数功能:根据加速功能设置加速.如果加速没有被导入,那么将会被导入.该调用将会被阻塞.脚本将等待直到加速完全导入为止.
- 函数参数:
  - speed:你希望填入的速度.范围必须在[1.0E-9;1.0E9]
- 函数返回值: 成功返回真,失败返回error.

setValues

- 函数原型

```
mixed setValues  (  table   values )
```

- 函数参数
  - values:表类型,该表的元素也是表类型,元素包含三个关键字:address (long), value (string with a value), flags (one of the constants TYPE_*).
- 返回值:true或者字符串error
- 例子代码:

```
gg.searchNumber('10', gg.TYPE_DWORD)
local r = gg.getResults(5) -- load items
r[1].value = '15'
print('Edited: ', gg.setValues(r))
local t = {}
t[1] = {}
t[1].address = 0x18004030 -- some desired address
t[1].flags = gg.TYPE_DWORD
t[1].value = 12345
t[2] = {}
t[2].address = 0x18004040 -- another desired address
t[2].flags = gg.TYPE_BYTE
t[2].value = '7Fh'
print('Set', t, gg.setValues(t))
```

setvisible

- 函数原型:

```
setVisible  (  bool   visible )
```

- 函数功能:打开或关闭GameGuardian的UI
- 函数参数:打开设置true,关闭设置false
- 函数返回值:无

skipRestoreState

- 函数原型:

```
skipRestoreState  (   )
```

- 函数功能:请不要在脚本执行完毕后存储GameGuardian的状态.例如,默认情况下,一组内存单元在脚本执行完毕过后保存,该函数调用就会阻止保存.
- 函数参数无
- 函数返回值无
- 代码例子:

```
gg.setRanges(bit32.bxor(gg.REGION_C_HEAP, gg.REGION_C_ALLOC, gg.REGION_ANONYMOUS))
-- do some things like search values
-- gg.skipRestoreState() -- if you uncomment this line -
-- memory ranges after end script stay same as we set in first line.
-- If not - it will be restored to state which be before script run.
```

sleep

- 函数原型

```
sleep  (  int   milliseconds )
```

- 函数功能:指定一个数值(毫秒数)会使得脚本睡眠,以系统定时器和调度器的精确性和准确性为前提
- 函数参数:
  ` milliseconds:指定睡眠毫秒数.
- 返回值:无
- 例子代码:

```
-- 200 ms
gg.sleep(200)
-- 300 ms
local v = 300
gg.sleep(v)
```

startFuzzy

- 函数原型:

```
mixed startFuzzy  (  int   type = gg.TYPE_AUTO,
 long   memoryFrom = 0,
 long   memoryTo = -1
)
```

- 函数功能:指定具体参数,进行模糊搜索.
- 函数参数:
  - type:数据类型符号常量
  - memoryFrom: 搜索的内存单元起始地址
  - memoryTo:搜索的内存单元的结束地址.
- 函数返回值:成功True,失败"error"
- 例子代码:

```
gg.startFuzzy()
gg.startFuzzy(gg.TYPE_DWORD)
gg.startFuzzy(gg.TYPE_FLOAT)
gg.startFuzzy(gg.TYPE_BYTE, 0x9000, 0xA09000)
```

timejump

- 函数原型:

```
mixed timeJump  (  string   time )
```

- 函数功能: 时间跳跃
- 函数参数:
  - time:时间字符串,该字符串参数跟GameGuardian中时间跳跃的时间格式一致.
- 函数返回值:成功为true,失败返回字符串error
- 例子代码

```
print('jump 1:', gg.timeJump('42345678'))
-- jump for 1 year 125 days 2 hours 41 minutes 18 seconds
print('jump 2:', gg.timeJump('1:125:2:41:18'))
-- same as above
print('jump 3:', gg.timeJump('5:13'))
-- jump for 5 minutes 13 seconds
print('jump 4:', gg.timeJump('7:3:1'))
-- jump for 7 hours 3 minutes 1 seconds
print('jump 5:', gg.timeJump('3600'))
-- jump for 1 hour
print('jump 6:', gg.timeJump('2:15:54:32'))
-- jump for 2 days 15 hours 54 minutes 32 seconds
print('jump 7:', gg.timeJump('3600.15'))
-- jump for 1 hour 0.15 seconds
print('jump 8:', gg.timeJump('7:3:1.519'))
-- jump for 7 hours 3 minutes 1.519 seconds
```

toast

- 函数原型:

```
toast  (  string   text,
 bool   fast = false
)
```

- 函数功能:在屏幕底部显示一段信息.如果第二个参数设置为true,将显示短暂的显示信息,如果
- 函数参数
  - text: 显示文本提示信息.
  - fast:短时间内,显示一段提示信息
- 例子代码:

```
gg.toast('This is toast')
-- Show text notification for a long period of time
gg.toast('This is toast', true)
-- Show text notification for a short period of time
```

gg符号常量

BUILD

- 数据类型:int
- 数据意义:由GameGuardian建立的数字

CACHE_DIR

- 数据类型:string
- 数据意义:在文件系统中,给GameGuardian指定的缓存目录的绝对路径.当设备运行,且存储空间不足时,首先被删除的文件。且无法保证这些文件何时被删除。
- 备注:不应该依赖于系统去自动删除这些文件. 你们应该总是有充足的理由去设置它,比如将你缓存文件大小设置为1MB,并在超过你分配的空间时,删除这些文件.如果你的应用需要更大的存储空间,你们应该使用 EXT_CACHE_DIR 代替.这部分数据替换到内存,不要让其它app可见,也不要让用户可以篡改.

EXT_CACHE_DIR

- 数据类型:string
- 数据意义:存储在 shared/external 存储设备的绝对目录路径,GameGuardian能够支持持久化存储数据.

FILES_DIR

- 数据类型:string
- 数据意义:控制GameGuardian文件的目录路径.,放置在内存中,对其它app不可见.

LOAD_APPEND

- 数据类型: int
- 数据意义: 用于loadLIst的标志,附加到列表.

LOAD_VALUES

- 数据类型:int
- 数据意义:用于loadLIst的标志,导入值

LOAD_VALUES_FREEZE

- 数据类型:int
- 数据意义:用于loadLIst的标志,导入值并冻结

PACKAGE

- 数据类型:string
- 数据意义:GameGuardian的包名字.

REGION_ANONYMOUS

- 数据类型:int

- 数据意义:用于getRanges, setRanges. 的标志位.

REGION_ASHMEM

- 数据类型:int
- 数据意义:用于getRanges, setRanges. 的标志位.

REGION_BAD

- 数据类型:int
- 数据意义:用于getRanges, setRanges. 的标志位.

REGION_C_ALLOC

- 数据类型:int
- 数据意义:用于getRanges, setRanges. 的标志位.

REGION_C_BSS

- 数据类型:int
- 数据意义:用于getRanges, setRanges. 的标志位.

REGION_C_DATA

- 数据类型:int
- 数据意义:用于getRanges, setRanges. 的标志位.

REREGION_C_HEAP

- 数据类型:int
- 数据意义:用于getRanges, setRanges. 的标志位.

REGION_CODE_APP

- 数据类型:int
- 数据意义:用于getRanges, setRanges. 的标志位.

REGION_CODE_SYS

- 数据类型:int
- 数据意义:用于getRanges, setRanges. 的标志位.

REGION_JAVA

- 数据类型:int
- 数据意义:用于getRanges, setRanges. 的标志位.

REGION_JAVA_HEAP

- 数据类型:int
- 数据意义:用于getRanges, setRanges. 的标志位.

REGION_OTHER

- 数据类型:int
- 数据意义:用于getRanges, setRanges. 的标志位.

REGION_PPSSPP

- 数据类型:int
- 数据意义:用于getRanges, setRanges. 的标志位.

REGION_STACK

- 数据类型:int
- 数据意义:用于getRanges, setRanges. 的标志位.

REGION_STACK

- 数据类型:int
- 数据意义:用于getRanges, setRanges. 的标志位.

SAVE_AS_TEXT

- 数据类型:int
- 数据意义:用于svaeList的标志位.

SIGN_EQUAL

- 数据类型:int
- 数据意义:用于searchAddress, searchNumber.的标志位.

SIGN_FUZZY_EQUAL

- 数据类型:int
- 数据意义:用于searchFuzzy.的标志位.

SIGN_FUZZY_GREATER

- 数据类型:int
- 数据意义:用于searchFuzzy.的标志位.

SIGN_FUZZY_LESS

- 数据类型:int
- 数据意义:用于searchFuzzy.的标志位.

SIGN_GREATER_OR_EQUAL

- 数据类型:int
- 数据意义:用于searchAddress, searchNumber的标志位.

SIGN_LESS_OR_EQUAL

- 数据类型:int
- 数据意义:用于searchAddress, searchNumber的标志位.

SIGN_NOT_EQUAL

- 数据类型:int
- 数据意义:用于searchAddress, searchNumber的标志位.

TYPR_AUTO

- 数据类型: int
- 数据意义:表示数据类型Auto

TYPR_BYTE

- 数据类型: int
- 数据意义:表示数据类型BYTE

TYPR_DOUBLE

- 数据类型: int
- 数据意义:表示数据类型DOUBLE

TYPR_DWORD

- 数据类型: int
- 数据意义:表示数据类型DWORD

TYPR_FLOAT

- 数据类型: int
- 数据意义:表示数据类型FLOAT

TYPR_QWORD

- 数据类型: int
- 数据意义:表示数据类型QWORD

TYPR_WORD

- 数据类型: int
- 数据意义:表示数据类型WORD

TYPR_XOR

- 数据类型: int
- 数据意义:表示数据类型XOR

VERSION

- 数据类型: string
- 数据意义: GameGuardian的版本号字符串.

VERSION_INT

- 数据类型: int
- 数据意义: GameGuardian的版本数字.