项目名称：Kooboo.AppMonitor

简介：是通过配置文件在Linux下来进行对程序进行监控的控制台程序。目前已包含的主要功能有：自动监控应用状态，获取某一时刻的线程栈和GC堆快照，进程的kill后重启守护，进程崩溃时的状态捕捉，日志记录。

启动该程序有2种 方式。

1：命令行方式

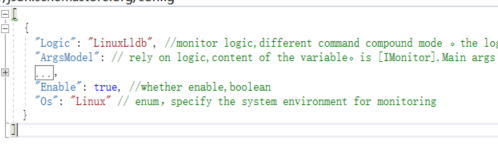
2：配置文件方式

1. 命令行方式，是属于一次性消费，可支持的命令目前有：

-p [pid] ： 对当前pid进程直接进行转储分析，在程序分析完后提供手动分析功能，转储文件和日志均在根目录。

- f [path] : 对path位置处的转储文件进行分析，在程序分析完后提供手动分析功能，日志在path处所在位置。

1. 配置文件方式，是通过配置文件的配置来开启自动化监控分析流程，配置文件样例如下：



首先，该配置文件最外层是以数组的方式来表达，表达为逻辑集合。数组中的每个对象，称之为单个的逻辑对象，逻辑对象为固定格式，共有四个键：Logic(逻辑对象的名称)、ArgsModel(该逻辑对象所需要的参数)、Enable(是否启动该逻辑对象)、Os(该逻辑对象的运行平台)。

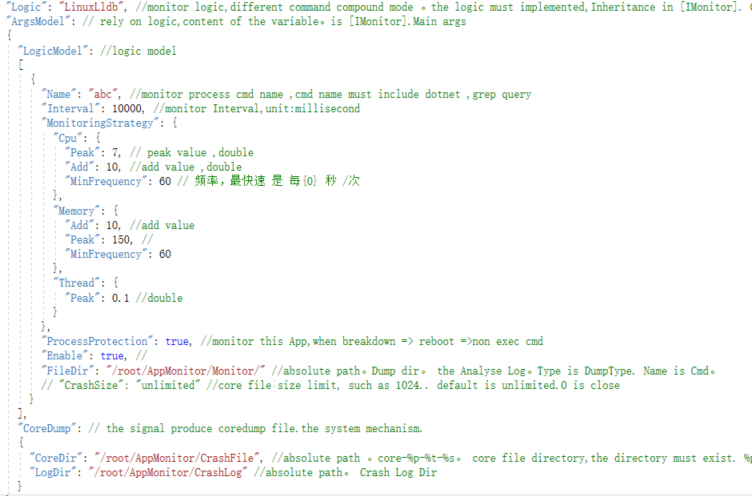
在程序以配置文件运行的时候，程序首先会判断当前平台系统，对每个（Enable为ture，并且Os满足当前平台的）逻辑对象进行创建。创建的对象名称为Logic属性值，在创建完成之后会调用该对象的Main(object o)方法并将ArgsModel属性的值当作参数传进去。

值得注意的是，本次反射创建逻辑对象并调用的环节，走的是接口方式调用(实现IMonitor接口)，而不是约定大于配置方式。另一方面，通过该流程可以见得这个逻辑对象不仅需要在配置文件中配置，还需要在代码中进行实现，也就是说你要实现Logic对象，不过具体实现没有进行其它要求，既没有规定ArgsModel属性的格式，也没有通过代码限制Logic Main()方法的后续调用的逻辑，但不能离开主题，自动化监控分析。

目前，现有代码实现的逻辑对象为：LinuxLldb，日后可以因为不同功能的扩展再实现其它的逻辑对象，如LinuxGdb、WindowsDbg等，可以一套代码同时运行不同的监控流程体系。

LinuxLldb，是以通过调用ps命令来实施的监控策略，是以.NET Core 2 SDK – createdump为转储工具，是以LLDB为分析工具，是以SOS为辅助分析工具，故而，在LinuxLldb运行初期会做一个简单的环境检查。

目前，LinuxLldb拥有的参数属性有2个：LogicModel，一个数组，会对数组中定义的多个进程对象进行监控。CoreDump，一个对象，用来定义全局的崩溃文件和日志的位置。



在LogicModel数组中的被监控对象有6个属性：

Name定义了被监控对象的cmd（在linux下，进程的名称表现为其打开进程的命令），检测到包含该Name的第一个进程对象进行监控。

Interval定义了每一次监控的间隔，以毫秒为单位。

MonitoringStrategy定义了监控的策略，会对CPU的占用率和内存的占用率进行监控，Peak为峰值，Add为增值，满足这2个任何一个并且间隔时间大于MinFrequency{秒}即会对其进行转储分析，Thread为该进程要进行分析的线程，目前只有一个属性peak，即对线程占用率大于peak的进行分析。

ProcessProtection为进程kill掉的重启守护开关，如果为ture那么进程被关闭就会自动重启。

Enable为是否启动该进程对象的监控。

FileDir为文件保存日志的目录，文件有2种：转储文件，转储文件日志，在其目录下又会划分为CpuPeak, CpuAdd, RamPeak, RamAdd这4个子目录。

LinuxLldb对于进程的崩溃转储是属于默认的开启行为，不过得依赖一个条件：即必须使用dumpexec(脚本位于bin目录下，./dumpexec “dotnet kooboo.dll”)打开应用程序。

LinuxLldb对于进程的监控分析会输出在控制台上，结构为 1.监控状态显示 2.转储时间计量 3.分析堆栈情况并对托管代码位置处予以代码显示(需要pdb文件位于dll同一目录下) 4.对堆对象类型的一个统计 5.对上一个快照堆的对比

经过测量，堆kooboo.server.dll的转储耗费时间超过6秒，在转储期间kooboo.server.dll的所有线程将会被挂起，也就是说kooboo要面临超过6秒的卡死状态。

转储器的机制就是在某一时刻分析线程栈，既然是某一时刻那么就必须挂起所有线程，以免线程栈段内容变化(读写冲突)，固所有的转储器都会使应用程序卡死。

使用fork快照一个相同进程的方式，因无法控制fork后的子进程状态，固该方法也有待证实。

那么如何让线程不挂死的问题就变成了如何转储时间耗费尽可能少？

1. 采用将转储内容缓存到内存区就恢复线程执行的转储器，不使用将转储内容落地到磁盘后才恢复线程执行的转储器。
2. 仅仅转储线程堆栈段。