



(12)发明专利

(10)授权公告号 CN 103268287 B

(45)授权公告日 2017.03.15

(21)申请号 201310222691.5

(22)申请日 2013.06.05

(65)同一申请的已公布的文献号
申请公布号 CN 103268287 A

(43)申请公布日 2013.08.28

(73)专利权人 福州瑞芯微电子股份有限公司
地址 350000 福建省福州市鼓楼区软件大道89号18号楼

(72)发明人 张维

(74)专利代理机构 福州市景弘专利代理事务所
(普通合伙) 35219
代理人 林祥翔 吕元辉

(51)Int.Cl.
G06F 11/36(2006.01)

(56)对比文件

CN 101539870 A,2009.09.23,说明书5-7,9-11页.

US 7823021 B2,2010.10.26,全文.

审查员 田晶

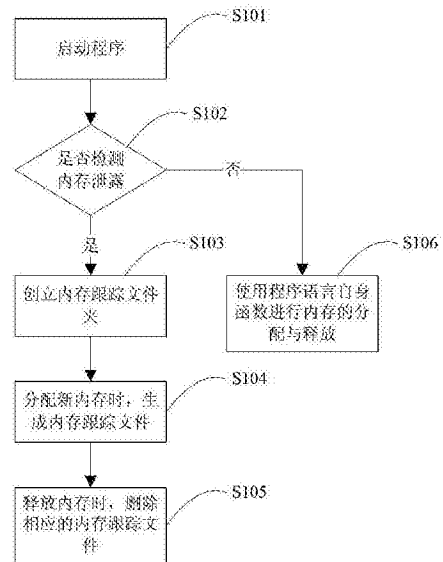
权利要求书1页 说明书4页 附图2页

(54)发明名称

检测内存泄露的方法与装置

(57)摘要

本发明提供一种检测内存泄露的方法与装置,既可以直观方便地检测内存泄露,又不会使生成的目标代码过大,也可以非常方便的切换检测开关。包括以下步骤:启动程序;分配新内存时,根据内存分配的地址,在设定的路径内生成内存跟踪文件,内存跟踪文件的内容或文件名与分配的内存地址、分配内存的文件或分配内存的源代码的行号相关;释放内存时,根据传进的内存地址,删除相应的内存跟踪文件。当程序运行完毕之后,可以根据设定的路径内所存在的内存跟踪文件判断内存泄露的情况。若设定的路径内存在未删除的内存跟踪文件,则说明有分配的内存未被释放。



1. 一种检测内存泄露的方法,包括以下步骤:

启动程序;

分配新内存时,根据内存分配的地址,在设定的路径内生成内存跟踪文件,内存跟踪文件的内容或文件名与分配的内存地址、分配内存的文件或分配内存的源代码的行号相关;

释放内存时,根据传进的内存地址,删除相应的内存跟踪文件;

若不检测内存泄露,使用程序语言自身函数进行内存的分配与释放;

判断是否检测内存泄露是通过检测配置文件中检测开关是否开启进行判断;

在步骤启动程序与步骤分配新内存之间,进行步骤:

创立内存跟踪文件夹;

在步骤分配新内存时,根据内存分配的地址,在内存跟踪文件夹内生成内存跟踪文件。

2. 根据权利要求1所述的检测内存泄露的方法,其特征在于,所述内存跟踪文件的文件名为分配的内存的地址号,文件内容为分配内存的文件和分配内存的源代码的行号。

3. 根据权利要求1所述的检测内存泄露的方法,其特征在于,在步骤启动程序之后,判断是否检测内存泄露,若是,则进行步骤:根据内存分配的地址,在设定的路径内生成内存跟踪文件;若否,不检测内存泄露。

4. 根据权利要求1至3任意一项所述的检测内存泄露的方法,其特征在于,

分配新内存通过运行定义的分配内存宏实现,所述分配内存宏与以下参数相关:要分配的内存的地址、代码文件的名称以及程序代码的行号;所述释放内存是通过定义的释放内存宏实现的,所述释放内存与参数:分配的内存的地址相关。

5. 根据权利要求1至3任意一项所述的检测内存泄露的方法,其特征在于,所述程序的源代码使用C,C+,C++或OBJECTIVE-C语言编写。

6. 一种检测内存泄露的装置,其特征在于,所述检测内存泄露的装置包括以下模块:

生成模块,分配新内存时,根据内存分配的地址,在设定的路径内生成内存跟踪文件,内存跟踪文件的内容或文件名与分配的内存地址、分配内存的文件或分配内存的源代码的行号相关;并通过检测配置文件中检测开关是否开启进行判断;

删除模块,用于释放内存时,根据传进的内存地址,删除相应的内存跟踪文件;

所述生成模块还用于创立内存跟踪文件夹,在步骤分配新内存时,根据内存分配的地址,在内存跟踪文件夹内生成内存跟踪文件,所述内存跟踪文件的文件名为分配的内存的地址号,文件内容为分配内存的文件和分配内存的源代码的行号。

检测内存泄露的方法与装置

技术领域

[0001] 本发明涉及计算机领域,尤其涉及一种检测计算机程序运行过程中内存泄露的方法与装置。

背景技术

[0002] 在使用支持有内存分配的语言,(例如C,C+,C++,OBJECTIVE-C语言)过程中,需要大量的手动分配内存,而分配之后的释放时常会遗忘,造成内存泄漏,因此对于内存泄漏的检测是十分必要的。

[0003] 目前,业界已经有许多代码和工具用于做内存泄漏的检测,但是通过代码运行后进行内存泄漏检测,其检测后的内容不直观,还需要分析是哪一行的代码存在内存泄漏。如果使用工具进行检测,消耗的成本比较大,而且可能会产生漏测的情况。而且通常的测量方式存在生成的目标代码过大,检测开关切换不方便等问题。

发明内容

[0004] 本发明的目的在于,提供一种检测内存泄露的方法与装置,既可以直观方便地检测内存泄露,又不会使生成的目标代码过大,也可以非常方便的切换检测开关。

[0005] 为实现上述发明目的,本发明提供了一种检测内存泄露的方法,包括以下步骤:

[0006] 启动程序;

[0007] 分配新内存时,根据内存分配的地址,在设定的路径内生成内存跟踪文件,内存跟踪文件的内容或文件名与分配的内存地址、分配内存的文件或分配内存的源代码的行号相关;

[0008] 释放内存时,根据传进的内存地址,删除相应的内存跟踪文件。

[0009] 优选地,在步骤启动程序与步骤分配新内存之间,进行步骤:

[0010] 创立内存跟踪文件夹;

[0011] 在步骤分配新内存时,根据内存分配的地址,在内存跟踪文件夹内生成内存跟踪文件。

[0012] 优选地,所述内存跟踪文件的文件名为分配的内存的地址号,文件内容为分配内存的文件和分配内存的源代码的行号。

[0013] 优选地,在步骤启动程序之后,判断是否检测内存泄露,若是,则进行步骤:根据内存分配的地址,在设定的路径内生成内存跟踪文件;若否,不检测内存泄露。

[0014] 更优选地,若不检测内存泄露,使用程序语言自身函数进行内存的分配与释放。

[0015] 优选地,分配新内存通过运行定义的分配内存宏实现,所述分配内存宏与以下参数相关:要分配的内存的地址、代码文件的名称以及程序代码的行号;所述释放内存是通过定义的释放内存宏实现的,所述释放内存与参数:分配的内存的地址相关。

[0016] 更优选地,所述判断是否检测内存泄露是通过检测配置文件中检测开关是否开启进行判断。

[0017] 优选地,所述程序的源代码使用C,C+,C++或OBJECTIVE-C语言编写。

[0018] 为实现本发明的另一个发明目的,本发明提供了一种检测内存泄露的装置,所述检测内存泄露的装置包括以下模块:

[0019] 生成模块,分配新内存时,根据内存分配的地址,在设定的路径内生成内存跟踪文件,内存跟踪文件的内容或文件名与分配的内存地址、分配内存的文件或分配内存的源代码的行号相关;

[0020] 删除模块,用于释放内存时,根据传进的内存地址,删除相应的内存跟踪文件。

[0021] 优选地,所述生成模块还用于创立内存跟踪文件夹,在步骤分配新内存时,根据内存分配的地址,在内存跟踪文件夹内生成内存跟踪文件,所述内存跟踪文件的文件名为分配的内存的地址号,文件内容为分配内存的文件和分配内存的源代码的行号。

[0022] 与现有技术不同地,本发明通过分配新内存时,根据内存分配的地址,在设定的路径内生成内存跟踪文件,然后释放内存时,根据传进的内存地址,删除相应的内存跟踪文件。由于内存跟踪文件的内容或文件名与分配的内存地址、分配内存的文件或分配内存的源代码的行号相关;因此当程序运行完毕之后,可以根据设定的路径内所存在的内存跟踪文件判断内存泄露的情况。若设定的路径内存在未删除的内存跟踪文件,则说明有分配的内存未被释放。本发明操作十分方便,且不会使生成的目标代码过大,通过肉眼观察即可很直观地发现代码文件中是否出现内存泄漏的问题。而且本发明布署方便,无需工具,不需要开发人员具备很强的代码调试能力;同时不需要开发人员全程参与,减小测试时对开发人员的依赖。

附图说明

[0023] 图1为本发明具体实施方式所述检测内存泄露的方法的流程图;

[0024] 图2为本发明具体实施方式所述检测内存泄露的装置的模块图。

具体实施方式

[0025] 为详细说明本发明的技术内容、构造特征、所实现目的及效果,以下结合实施方式并配合附图详予说明。

[0026] 请参阅图1本实施例提供了一种检测内存泄露的方法,包括以下步骤:

[0027] S101 启动程序;

[0028] S102判断是否检测内存泄露,若是,则进行内存泄露检测;若否,不检测内存泄露,进行步骤S106使用程序语言自身函数进行内存的分配与释放。

[0029] 进行内存泄露检测的步骤具体如下,

[0030] S103创立内存跟踪文件夹;

[0031] S104分配新内存时,根据内存分配的地址,在内存跟踪文件夹内生成内存跟踪文件,内存跟踪文件的内容或文件名与分配的内存地址、分配内存的文件或分配内存的源代码的行号相关;在某些实施例中,所述内存跟踪文件的文件名为分配的内存的地址号,文件内容为分配内存的文件和分配内存的源代码的行号。这样可以更加方便直观地根据文件名以及文件中保存的内容得知哪个代码文件的哪一段程序代码分配的内存没有被释放,操作人员即可采取相应措施。

[0032] 在某些实施例中,可以采用与“创立内存跟踪文件夹,进而在后续分配内存时,将内存跟踪文件生成在创立的内存跟踪文件夹内”这一方案不同的技术方案。设定的路径可以是设定的文件夹,也可以是盘符下的根目录。例如用于生成内存跟踪文件的设定的文件夹可以不必是创立的内存跟踪文件夹,而直接指定一个现有的文件或指定的目标文件夹。

[0033] S105释放内存时,根据传进的内存地址,删除相应的内存跟踪文件。

[0034] 在结束程序的时候,可以手动或者使用工具来分析跟踪目录,以得到内存泄漏的信息。例如根据各文件中保存的内容进行统计形成一个内存泄露表,根据内存泄露表即可得知各个代码文件存在的内存泄露的严重程度。

[0035] 在某些实施例中,设计内存管理的函数如下:

[0036] `void* et_alloc(void **ppAddr,int iCount, char* szFile, int iLine);`

[0037] 这是分配内存的函数,

[0038] 在这个函数中,将分配内存,同时创建跟踪文件,记录行号和文件名,

[0039] ppAddr这个二级指针是将要分配内存的地址,

[0040] szFile是代码文件名,

[0041] iLine是行号,

[0042] `void et_free(void **ppAddr);`

[0043] 这是释放内存的函数,同时还要删除内存跟踪文件。

[0044] 分配新内存通过运行定义的分配内存宏实现,所述分配内存宏与以下参数相关:要分配的内存的地址、代码文件的名称以及程序代码的行号;所述释放内存是通过定义的释放内存宏实现的,所述释放内存与参数:分配的内存的地址相关。同时宏还用于获取文件名,代码的行号等参数,并把这些信息传给相应的函数,生成内存跟踪文件,或删除相应的内存跟踪文件。

[0045] 某些实施例中,文件标记不仅适用在跟踪文件,同时适用在配置中。

[0046] 在相应的情况下,需要在一套已经生成的二进制可执行文件中同时支持检测与不检测。

[0047] 可以把检测开关放在配置文件中,需要的时候打开检查。为方便检测开关的切换,在优选的实施例中,可以通过配置文件实现检测开关的切换,例如可以将检测开关放置在文件根目录下,需要切换检测与否时,找到检测文件,修改0或1(当然在具体实现时,有可能是true/false,也有可能是0/1,也有可能是on/off),就可以实现配置,无需重新编译。

[0048] 这适用在已经把可执行程序发布给使用者,然后又需要在不重新编译的情况下进行泄漏检查,

[0049] 做法就是在et_alloc和et_free这一套分配和释放的函数中,读取全局的一个变量,而这一全局变量,在程序启动时,就从配置文件中读取了。

[0050] 在程序启动时,对内存泄漏是否进行检测做一判定,内存泄漏是否进行检测的判定具体为:定义一配置文件,该配置文件设置有一是否进行检测的开关,该配置文件的内容配置为0时,则检测开关关闭,为1时,则检测开关开启。判断是否检测内存泄露是通过检测配置文件中检测开关是否开启进行判断。

[0051] 上述实施例所述程序的源代码使用C,C+,C++或OBJECTIVE-C语言编写。

[0052] 本发明的上述实施例可以适用于:c语言,c++语言(可以相应的扩展new,delete),

objective-c ,

[0053] 上述实施例具有如下优势:可以直观地发现程序中内存泄露的问题,并能迅速找到泄露的原因;

[0054] 布署方便,成本低廉;不需要开发人员全程参与,减小测试时对开发人员的依赖。还可以便于异地开发调试,只需要将测试后的相应内存跟踪文件夹或内存跟踪文件通过网络发送给分析、解决问题的人员即可。

[0055] 如图2所示,本发明还提供了一种检测内存泄露的装置20,所述检测内存泄露的装置包括以下模块:

[0056] 生成模块201,分配新内存时,根据内存分配的地址,在设定的路径内生成内存跟踪文件,内存跟踪文件的内容或文件名与分配的内存地址、分配内存的文件或分配内存的源代码的行号相关;在某些实施例中,所述生成模块还用于创立内存跟踪文件夹,在步骤分配新内存时,根据内存分配的地址,在内存跟踪文件夹内生成内存跟踪文件,所述内存跟踪文件的文件名为分配的内存的地址号,文件内容为分配内存的文件和分配内存的源代码的行号。

[0057] 删除模块202,用于释放内存时,根据传进的内存地址,删除相应的内存跟踪文件。

[0058] 以上所述仅为本发明的实施例,并非因此限制本发明的专利范围,凡是利用本发明说明书及附图内容所作的等效结构或等效流程变换,或直接或间接运用在其他相关的技术领域,均同理包括在本发明的专利保护范围内。

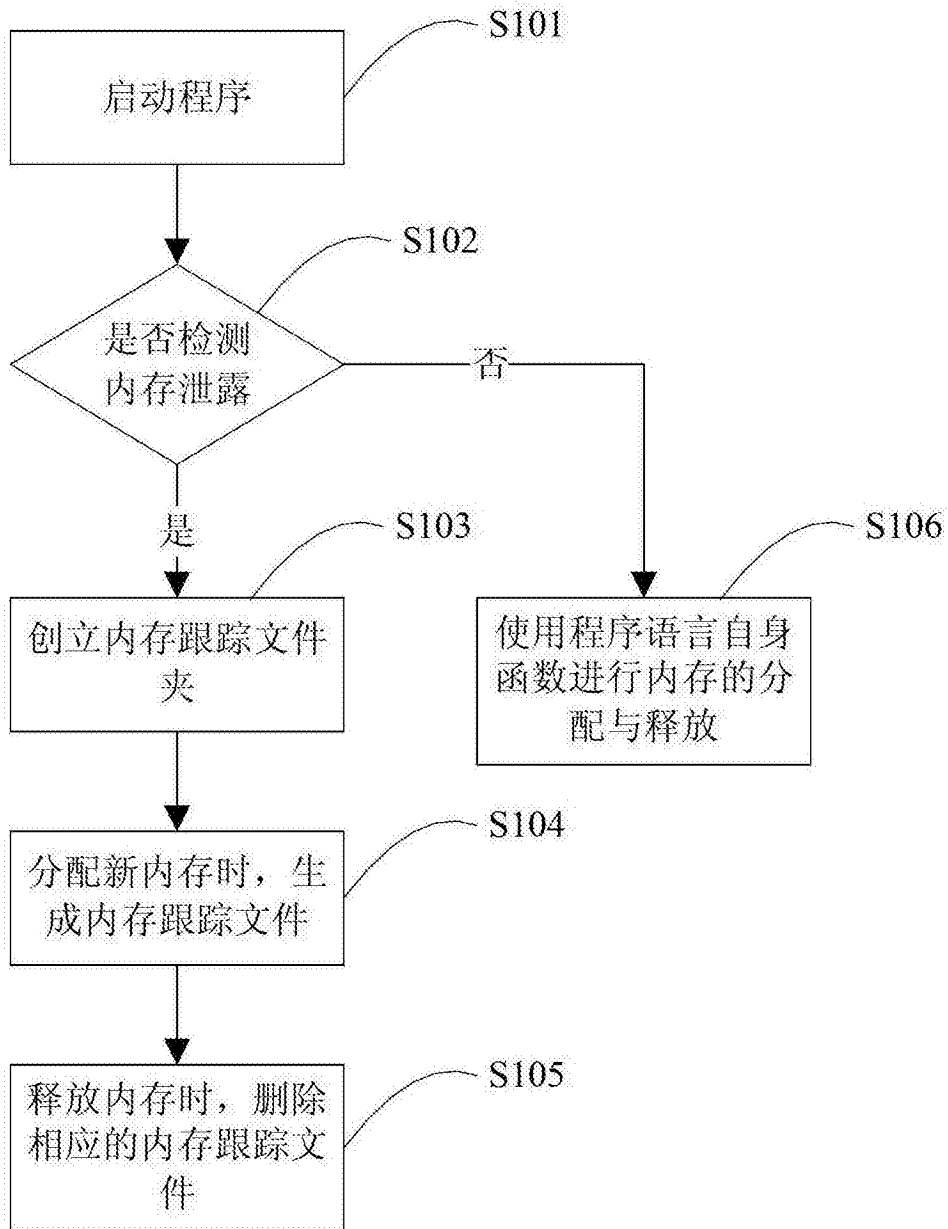


图1

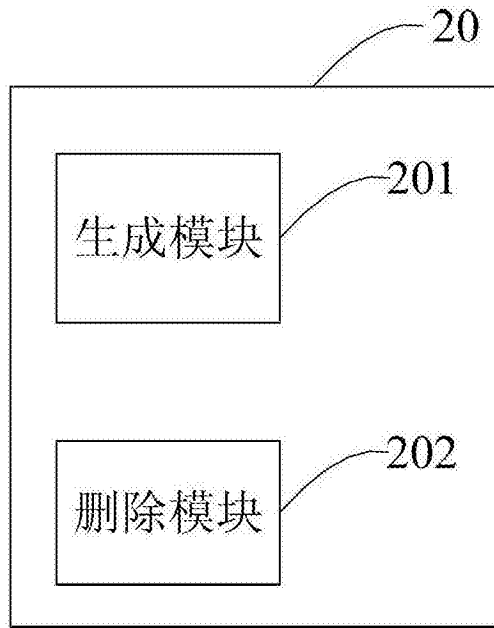


图2