权 利 要 求 书

1、一种模块生命周期管理方法，包括模块在操作系统中的装载过程和卸载过程，其特征在于，所述的装载过程包括以下步骤：

（1）应用程序向操作系统提出装载模块的请求；

（2）操作系统根据应用程序的请求查找系统全局资源，如果该模块不存在，则进行该模块对象的装载处理；如果该模块已存在，则进行增加该模块对象引用计数处理；

（3）操作系统根据该模块的依赖关系为应用程序创建相应的模块视图；

（4）操作系统向用户程序返回相应模块的接口指针。

所述的卸载过程包括以下步骤：

（1）应用程序向操作系统提出减少相应模块引用记数的请求；

（2）操作系统根据应用程序的请求进行相应的减少模块对象引用计数处理，如果该模块对象引用计数符合卸载条件，则删除模块视图，并进行模块对象的卸载处理。

2、根据权利要求 1 所述的模块生命周期管理方法，其特征在于，所述的模块对象的装载处理包括以下步骤：

（1）操作系统根据应用程序请求装载的模块名从存储系统中读取指定的模块对象，并将其载入系统；

（2）将该模块对象保存的引用记数初始设定为 1；

（3）若该模块不依赖引用其他模块，则完成该模块对象的装载处理；

（4）若该模块依赖引用其他模块，则操作系统查找系统全局资源，如果该被依赖模块不存在，则递归进行从所述步骤（1）开始的该被依赖模块对象的装载处理；如果该被依赖模块已存在，则进行增加该被依赖模块对象引用计数处理，直至所有被依赖模块均处理完毕，则完成该模块对象的装载处理。

3、根据权利要求 2 所述的模块生命周期管理方法，其特征在于，所述的存储系统为文件系统或者其他存储设备。

4、根据权利要求 1、2 或 3 所述的模块生命周期管理方法，其特征在于，所述的增加模块对象引用计数处理包括以下步骤：

（1）如果另一模块对象对该模块对象进行依赖引用，则将该模块对象所保存的引用计数增加 1；

（2）如果一应用程序建立模块视图对该模块对象进行引用，则将该模块对象所保存的引用计数增加 1。

5、根据权利要求 1 所述的模块生命周期管理方法，其特征在于，所述的模块对象的卸载处理包括以下步骤：

（1）将该模块对象从系统中卸载并回收其所占用的系统资源；

（2）若该模块不依赖引用其他模块，则完成该模块对象的卸载处理；

（3）若该模块依赖引用其他模块，则操作系统进行减少被依赖模块对象引用计数处理，并判断该被依赖模块对象引用计数是否符合卸载条件，如果符合卸载条件，则递归进行从所述步骤（1）开始的该被依赖模块对象的卸载处理，直至所有被依赖模块均处理完毕，则完成该模块对象的卸载处理。

6、根据权利要求 1 或 5 所述的模块生命周期管理方法，其特征在于，所述的模块对象卸载条件为该模块对象保存的的引用计数为 0。

7、根据权利要求 1 或 5 所述的模块生命周期管理方法，其特征在于，所述的减少模块对象引用计数处理包括以下步骤：

（1）如果另一模块对象取消对该模块对象的依赖引用，则将该模块对象所保存的引用计数减少 1；

（2）如果一应用程序删除模块视图取消对该模块对象的引用，则将该模块对象所保存的引用计数减少 1。

8、根据权利要求 6 所述的模块生命周期管理方法，其特征在于，所述的减少模块对象引用计数处理包括以下步骤：

（1）如果另一模块对象取消对该模块对象的依赖引用，则将该模块对象所保存的引用计数减少 1；

（2）如果一应用程序删除模块视图取消对该模块对象的引用，则将该模块对象所保存的引用计数减少 1。

9、根据权利要求 1 所述的模块生命周期管理方法，其特征在于，所述的模块为操作系统中的可执行文件或者动态链接库。

说　明　书

模块生命周期管理方法

技术领域

本发明涉及计算机操作系统领域，特别涉及操作系统模块管理领域，具体是指一种模块生命周期管理方法。

背景技术

模块，在计算机操作系统中均指可执行映像文件（包括可执行文件和动态库）在操作系统中的表示形式。

传统操作系统模块的生命周期是由模块装载器控制的，例如微软是通过 LoadLibrary API 进行模块的装载，通过 FreeLibrary API 进行模块的卸载。这些实现方法都需要用户主动调用系统 API 卸载模块。这将导致无法引用另一进程的模块资源，因为无法得知那个进程的模块何时被卸载。另外，如果模块指针传递给多个用户后 ，将很难决定该何时卸载该模块。如果是提前卸载，会导致正在使用该模块的用户出错；如果是最后卸载，会使得没有用户使用时仍占用内存等资源。

发明内容

本发明的目的是克服了上述现有技术中的缺点，提供一种基于引用计数的模块生命周期管理方法，可以进行一次性装载，对外部引用可进行自动计数控制，并在最终无外部引用时安全卸载。

为了实现上述的目的，本发明的模块生命周期管理方法如下：

该模块生命周期管理方法，包括模块在操作系统中的装载过程和卸载过程，其主要特点是，所述的装载过程包括以下步骤：

（1）应用程序向操作系统提出装载模块的请求；

（2）操作系统根据应用程序的请求查找系统全局资源，如果该模块不存在，则进行该模块对象的装载处理；如果该模块已存在，则进行增加该模块对象引用计数处理；

（3）操作系统根据该模块的依赖关系为应用程序创建相应的模块视图；

（4）操作系统向用户程序返回相应模块的接口指针。

所述的卸载过程包括以下步骤：

（1）应用程序向操作系统提出减少相应模块引用记数的请求；

（2）操作系统根据应用程序的请求进行相应的减少模块对象引用计数处理，如果该模块对象引用计数符合卸载条件，则删除模块视图，并进行模块对象的卸载处理。

该模块生命周期管理方法的模块对象的装载处理包括以下步骤：

（1）操作系统根据应用程序请求装载的模块名从存储系统中读取指定的模块对象，并将其载入系统；

（2）将该模块对象保存的引用记数初始设定为 1；

（3）若该模块不依赖引用其他模块，则完成该模块对象的装载处理；

（4）若该模块依赖引用其他模块，则操作系统查找系统全局资源，如果该被依赖模块不存在，则递归进行从所述步骤（1）开始的该被依赖模块对象的装载处理；如果该被依赖模块已存在，则进行增加该被依赖模块对象引用计数处理，直至所有被依赖模块均处理完毕，则完成该模块对象的装载处理。

该模块生命周期管理方法的存储系统可以为文件系统，也可以为其他存储设备。

该模块生命周期管理方法的增加模块对象引用计数处理包括以下步骤：

（1）如果另一个模块对象对该模块对象进行依赖引用，则将该模块对象所保存的引用计数增加 1；

（2）如果一个应用程序建立模块视图对该模块对象进行引用，则将该模块

对象所保存的引用计数增加 1。

该模块生命周期管理方法的模块对象的卸载处理包括以下步骤：

（1）将该模块对象从系统中卸载并回收其所占用的系统资源；

（2）若该模块不依赖引用其他模块，则完成该模块对象的卸载处理；

（3）若该模块依赖引用其他模块，则操作系统进行减少被依赖模块对象引用计数处理，并判断该被依赖模块对象引用计数是否符合卸载条件，如果符合卸载条件，则递归进行从所述步骤（1）开始的该被依赖模块对象的卸载处理，直至所有被依赖模块均处理完毕，则完成该模块对象的卸载处理。

该模块生命周期管理方法的模块对象卸载条件为该模块对象保存的的引用计数为 0。

该模块生命周期管理方法的减少模块对象引用计数处理包括以下步骤：

（1）如果另一个模块对象取消对该模块对象的依赖引用，则将该模块对象所保存的引用计数减少 1；

（2）如果一个应用程序删除模块视图取消对该模块对象的引用，则将该模块对象所保存的引用计数减少 1。

该模块生命周期管理方法的模块为操作系统中的可执行文件或者动态链接库。

采用了该发明的模块生命周期管理方法，由于模块是装载于全局空间中，并采用模块视图方式对其进行引用，因此只需一次性装载；同时由于该方法是基于引用计数的，因此模块只在没有任何引用时，才会被卸载，使得模块不会提前被卸载，并且在没有用户使用时会立即被卸载，同时还可允许引用其它的模块资源，从而有效地利用已有资源。

附图说明

图 1 为模块对象与模块视图关系示意图。

图 2 为本发明的模块装载过程的流程图。

图 3 为本发明的模块卸载过程的流程图。

图 4 为有依赖关系的模块对象之间及其与模块视图之间的关系示意图。

具体实施方式

为了能够更清楚地理解本发明的技术内容，特举以下实施例详细说明。

请参阅图 1 所示，模块生命周期管理中主要资源有两个：模块对象（Module），模块视图（Module View）。模块对象是系统内的全局资源，当进程加载模块时，需要引用全局的模块对象并把它映射到进程的虚拟地址空间中，我们把进程对模块对象的这种引用关系称为模块视图。

模块对象与模块视图是一对多的关系，一个模块可能同时被多个进程加载，但同一个模块对象在卸载之前实际只会被加载一次（第一次加载），以后的进程加载请求只是使模块对象的引用计数增加。但不同的进程在加载同一模块时都会在进程中创建各自的模块视图，这些模块视图保留了对模块对象的引用。

再请参阅图 2 所示，该图为模块装载过程，其中应用程序通过调用系统 API EzLoadModule()向操作系统提出装载模块的请求。操作系统根据请求装载的模块名从文件系统或其他设备中装载指定的模块，并返回该模块的接口指针。这时，模块的引用记数是 1。

如果把模块接口指针传给其他用户使用，则每传递一个用户就要对模块的引用记数加 1。当不再需要使用模块接口指针时，要减少一次模块的引用记数。

如果模块的引用记数为 0，就会进入模块卸载过程。

在装载过程中如有依赖其他模块，则会引起递归地装载所依赖的模块，并保留所依赖模块的一个引用记数（对所依赖模块的引用记数加 1）。

再请参阅图 3 所示，该图为模块卸载过程，其中，如果减少模块引用记数的时候发现本模块的引用记数为 0，就会进行模块卸载。

卸载过程大概可分两步：

（1）卸载本模块并回收系统资源；

（2）对所依赖的模块减少一次引用记数。

对于第（2）步（减少所依赖模块的引用记数），有可能导致连锁的模块卸载。

“和欣”操作系统是通过 EzLoadModule API 进行模块的装载，每多一个引用就对其引用记数加 1（调用 IModule 接口的 AddRef 方法），每少一个引用就对其引用记数减 1（调用 IModule 接口的 Release 方法）。当引用记数为 0 时，自动进行模块的卸载工作。

再请参阅图 4 所示，该图表明了这种模块间及模块与模块视图间的引用关系，模块对象的生命周期基于引用计数管理。一个模块对象可能被两方面引用到：一是模块视图，二是依赖于该模块对象的其它模块对象。

从图中可以看出，对于 c.dll 的模块对象，它的引用计数由模块对象 b.dll 以及模块视图 c.dll 保留，模块对象 b.dll 的引用计数由模块对象 a.dll 和模块视图 b.dll 保留，而模块对象 a.dll 的引用计数则由模块视图 a.dll 保留。

同时，模块视图之间也使用类似的引用计数关系，模块视图 b.dll 保留了模块视图 c.dll 的引用计数，模块视图 a.dll 保留了模块视图 b.dll 的引用计数，而模块视图 a.dll 的引用计数则由装载 a.dll 的请求者保留。

上述引用计数关系保证了模块对象及模块视图能够按照引用计数关系被正确的加载和卸载。当模块视图 a.dll 的引用计数为零时，它将释放掉模块视图 b.dll 以及模块对象 a.dll 的引用计数，从而在引用计数链上引起连锁引用计数反应，最终卸载掉整个模块链。

在此说明书中，本发明已参照其特定的实施例作了描述。但是，很显然仍可以作出各种修改和变换而不背离本发明的精神和范围。因此，说明书和附图应被认为是说明性的而非限制性的。

说　明　书　附　图

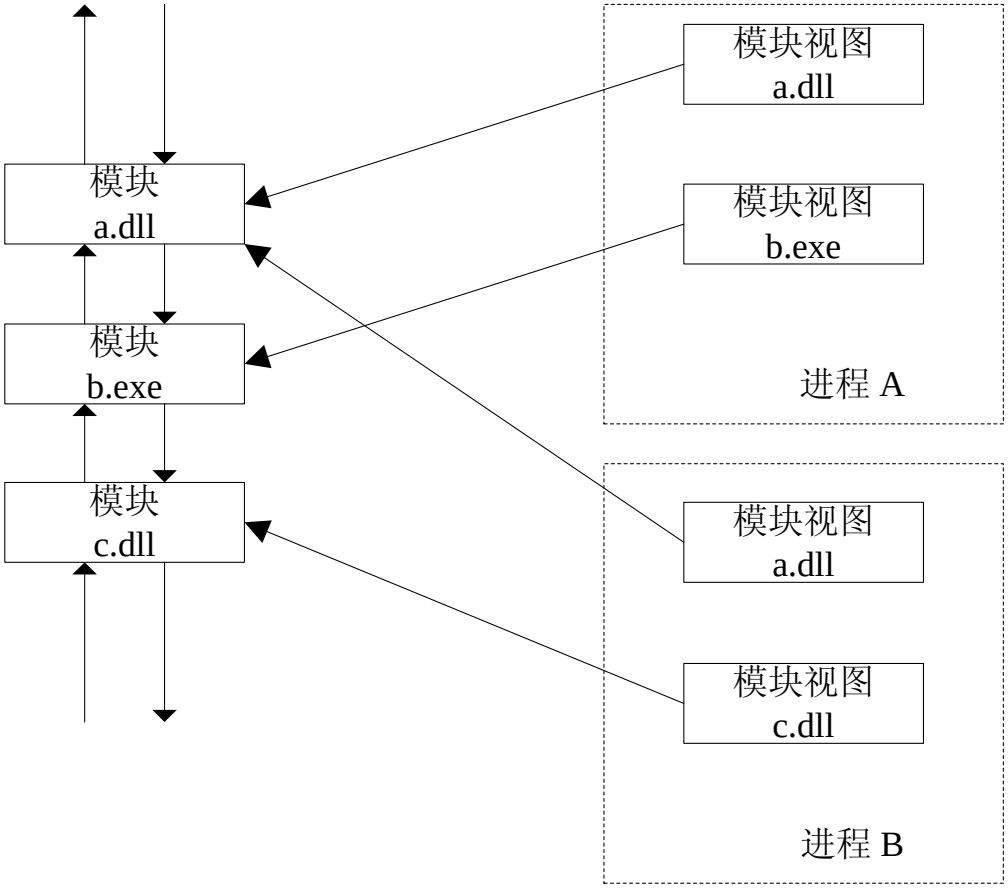

模块视图
a.dll
模块视图
b.exe
进程 A
模块
a.dll
模块
b.exe
模块
c.dll
模块视图
a.dll
模块视图
c.dll
进程 B

图 1

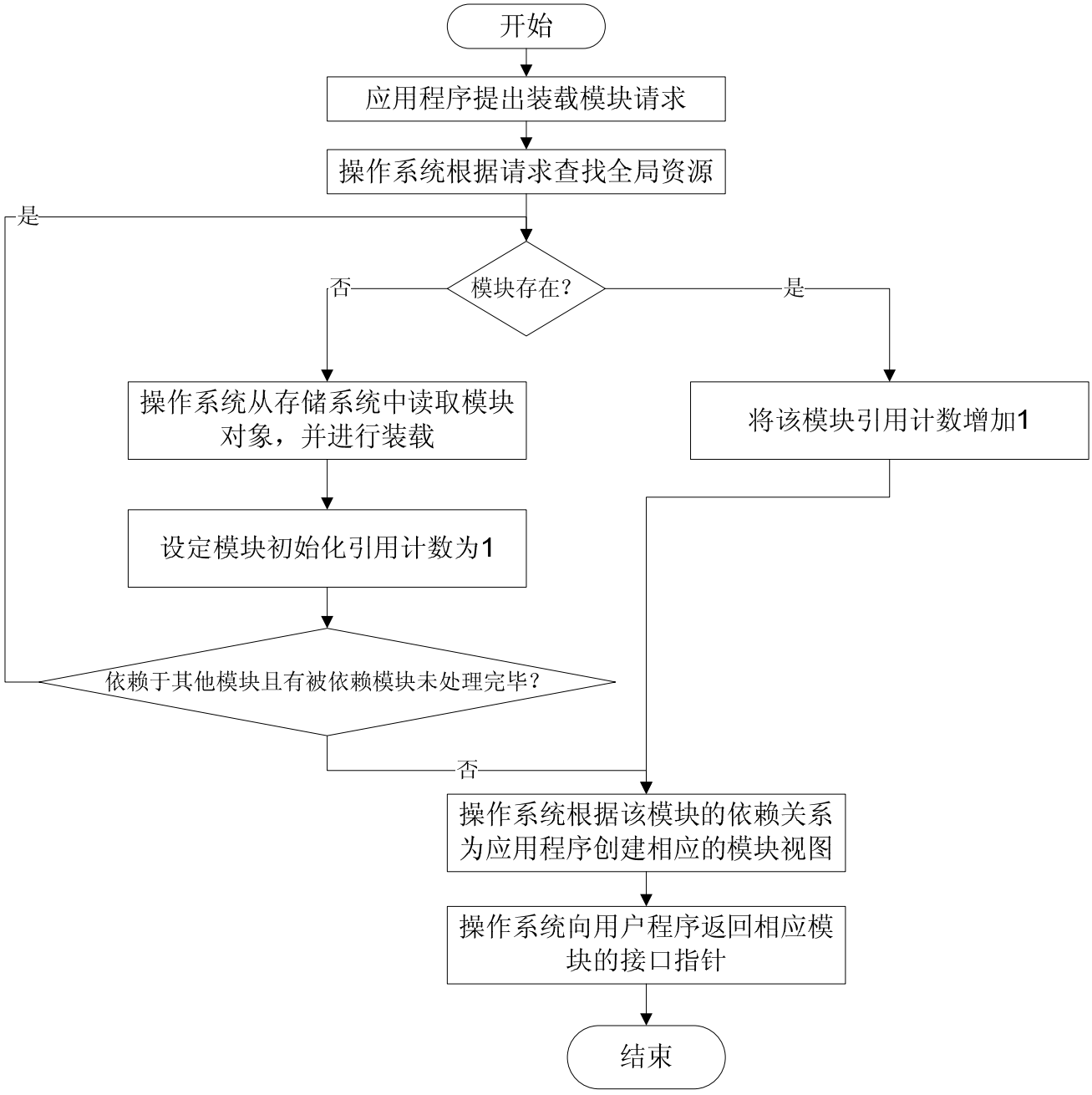

开始
应用程序提出装载模块请求
操作系统根据请求查找全局资源
是
模块存在?
否
是
操作系统从存储系统中读取模块对象，并进行装载
将该模块引用计数增加1
设定模块初始化引用计数为1
依赖于其他模块且有被依赖模块未处理完毕?
否
操作系统根据该模块的依赖关系为应用程序创建相应的模块视图
操作系统向用户程序返回相应模块的接口指针
结束

图 2

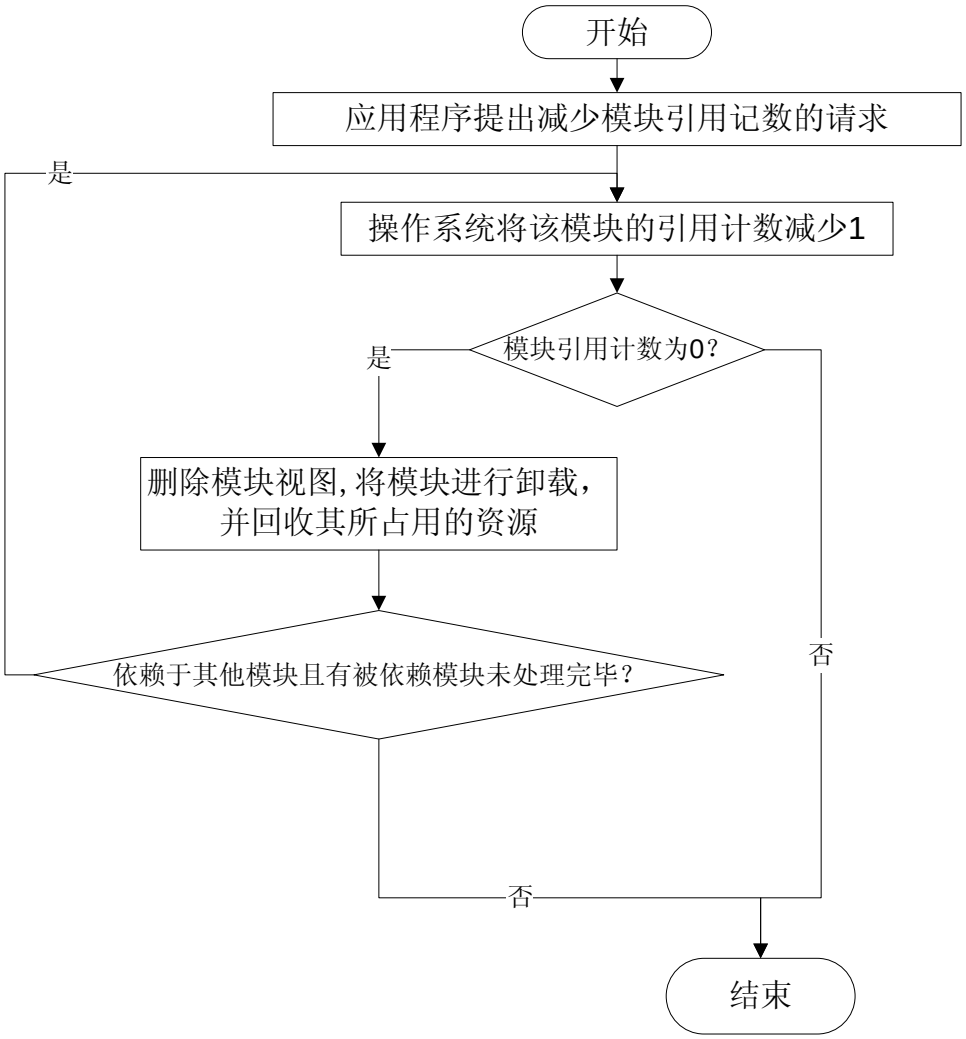

开始
应用程序提出减少模块引用记数的请求
操作系统将该模块的引用计数减少1
模块引用计数为0?
是
否
删除模块视图,将模块进行卸载,
并回收其所占用的资源
依赖于其他模块且有被依赖模块未处理完毕?
是
否
结束

图 3

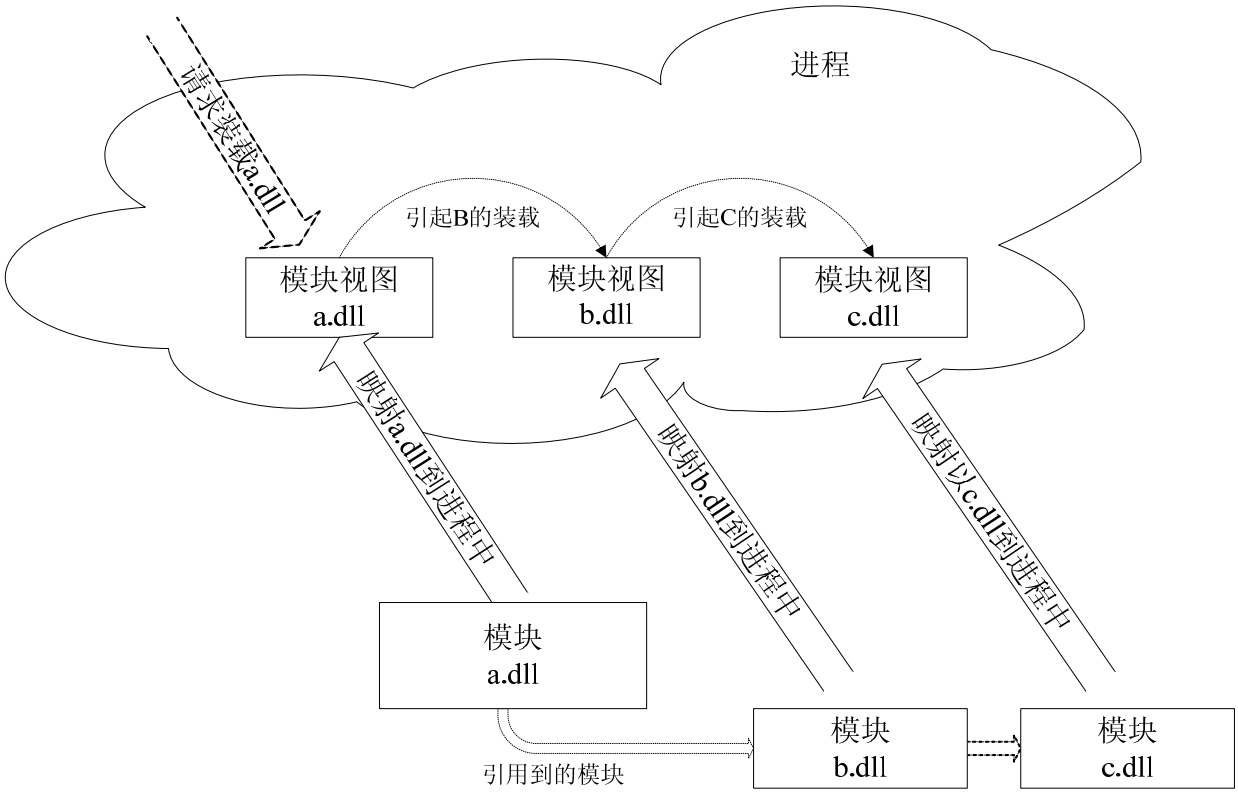
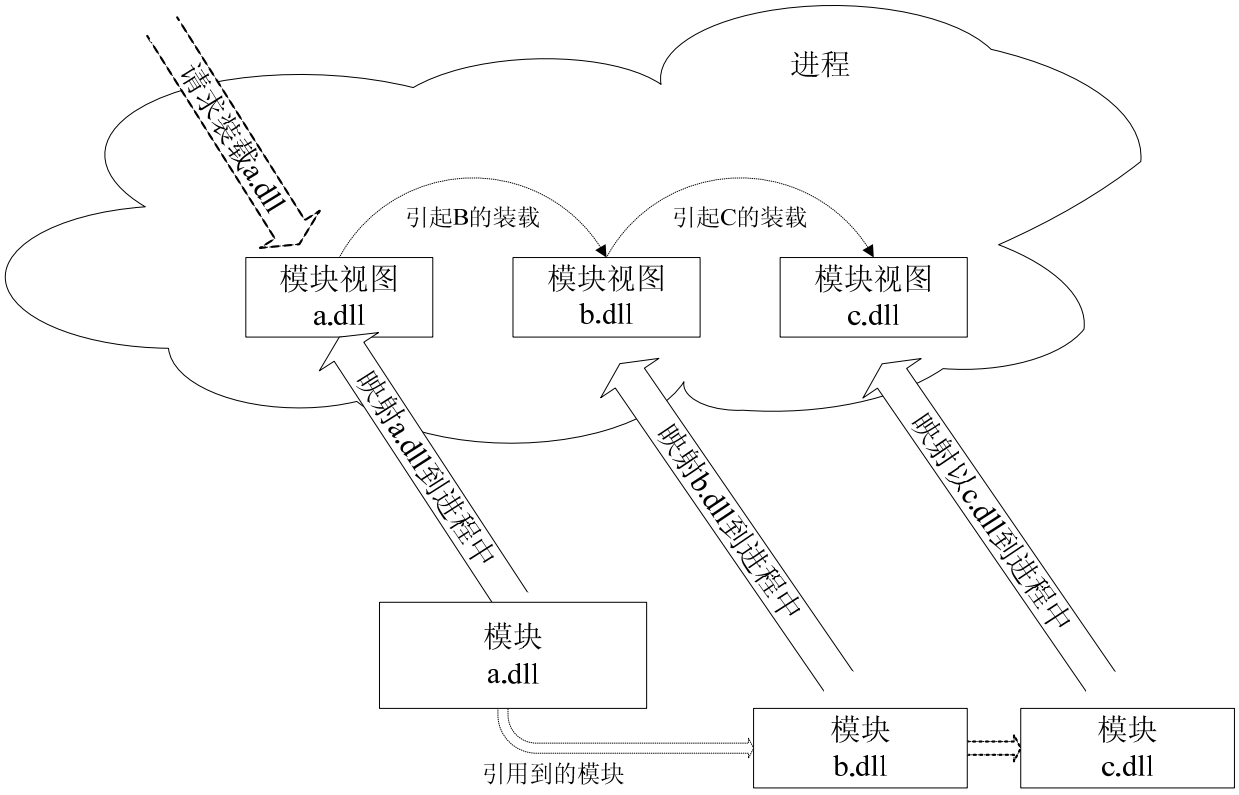

进程
请求装载a.dll
引起B的装载
引起C的装载
模块视图
a.dll
模块视图
b.dll
模块视图
c.dll
映射a.dll到进程中
映射b.dll到进程中
映射以c.dll到进程中
模块
a.dll
模块
b.dll
模块
c.dll
引用到的模块

图 4

说 明 书 摘 要

本发明涉及一种模块生命周期管理方法，包括模块装载过程和卸载过程，其中所述的装载过程包括应用程序提出装载模块请求、操作系统根据情况进行该模块对象的装载处理或者进行增加该模块对象引用计数处理、应用程序根据该模块的接口指针创建相应的模块视图；卸载过程包括应用程序删除模块视图并提出卸载模块请求、操作系统根据减少模块对象引用计数、根据情况进行模块对象的卸载处理。采用该种结构的模块生命周期管理方法，只需一次性装载；同时模块只在没有任何引用时才会被卸载，使得模块不会提前被卸载，并且在没有用户使用时会立即被卸载，同时还可允许引用其它的模块资源，从而有效地利用已有资源。

摘 要 附 图


进程
请求装载a.dll
引起B的装载
引起C的装载
模块视图
a.dll
模块视图
b.dll
模块视图
c.dll
映射a.dll到进程中
映射b.dll到进程中
映射以c.dll到进程中
模块
a.dll
模块
b.dll
模块
c.dll
引用到的模块