(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 104871173 B
(45)授权公告日 2018.01.16

(21)申请号 201380066213.8

(22)申请日 2013.12.20

(65)同一申请的已公布的文献号
申请公布号 CN 104871173 A

(43)申请公布日 2015.08.26

(30)优先权数据
2012-279085 2012.12.21 JP

(85)PCT国际申请进入国家阶段日
2015.06.17

(86)PCT国际申请的申请数据
PCT/JP2013/084351 2013.12.20

(87)PCT国际申请的公布数据
WO2014/098239 JA 2014.06.26

(73)专利权人 日本电信电话株式会社
地址 日本东京都

(72)发明人 秋山满昭 针生刚男

(74)专利代理机构 北京三友知识产权代理有限公司 11127
代理人 李辉 于靖帅

(51)Int.Cl.
G06F 21/54(2006.01)
G06F 21/14(2006.01)
G06F 21/55(2006.01)
G06F 21/56(2006.01)

(56)对比文件
US 2001014958 A1,2001.08.16,
CN 101739333 A,2010.06.16,
US 2002199172 A1,2002.12.26,
JP 2009031859 A,2009.02.12,
JP 2011028506 A,2011.02.10,
JP 2004185064 A,2004.07.02,
Takaaki MATSUMOTO 等."A Keylogger Detection Using Dynamic API Inspection".《Transactions of Information Processing Society of Japan》.2007,第48卷(第9期),3141-3142.

审查员 刘杰

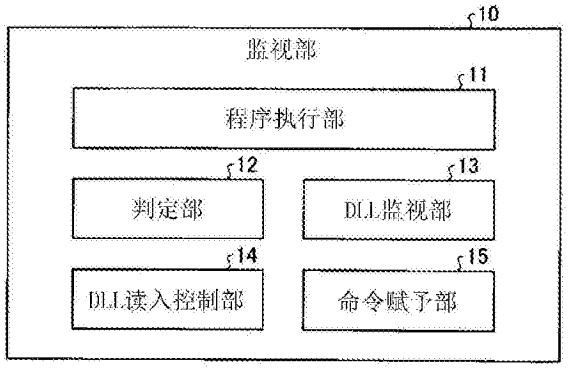
权利要求书1页 说明书8页 附图6页

(54)发明名称

监视装置和监视方法

(57)摘要

监视系统中的监视部(10)对所执行的程序是否是监视对象程序进行判定。然后,在判定为所执行的程序是监视对象程序的情况下,监视系统中的监视部(10)在包含于被该监视对象程序所调用的函数中的命令串之前按顺序赋予满足规定的条件的命令串、以及作为在满足了规定的条件的情况下使规定的控制处理开始的命令的条件分支命令。



CN 104871173 B

1.一种监视装置，其特征在于，该监视装置具有：

判定部，其对所执行的程序是否是监视对象程序进行判定；以及

命令赋予部，其在所述判定部判定为所执行的程序是监视对象程序的情况下，在每次加载安装有由该监视对象程序调用的钩挂对象API的DLL时，动态地生成第1命令串和条件分支命令，并在包含于所述钩挂对象API中的命令串之前按顺序赋予所述第1命令串和所述条件分支命令，所述第1命令串始终满足执行所述条件分支命令的条件，所述条件分支命令是在满足了所述条件的情况下使对所述监视对象程序的举动进行监视的处理开始的命令。

2.根据权利要求1所述的监视装置，其特征在于，

所述命令赋予部按顺序赋予进行运算或者比较以使存储于标志寄存器中的值满足所述条件的所述第1命令串、以及作为在所述标志寄存器满足了所述条件的情况下使对所述监视对象程序的举动进行监视的处理开始的命令的所述条件分支命令。

3.根据权利要求1所述的监视装置，其特征在于，

该监视装置还具有执行部，该执行部执行包含由所述命令赋予部赋予了条件分支命令和所述第1命令串的命令串在内的所述钩挂对象API，并且执行监视所述监视对象程序的举动的处理。

4.根据权利要求2所述的监视装置，其特征在于，

该监视装置还具有执行部，该执行部执行包含由所述命令赋予部赋予了条件分支命令和所述第1命令串的命令串在内的所述钩挂对象API，并且执行监视所述监视对象程序的举动的处理。

5.一种监视方法，该监视方法由监视装置执行，其特征在于，该监视方法包含：

判定步骤，对所执行的程序是否是监视对象程序进行判定；以及

命令赋予步骤，在由所述判定步骤判定为所执行的程序是监视对象程序的情况下，在每次加载安装有由该监视对象程序所调用的钩挂对象API的DLL时，动态地生成第1命令串和条件分支命令，并在包含于所述钩挂对象API中的命令串之前按顺序赋予所述第1命令串和所述条件分支命令，所述第1命令串始终满足执行条件分支命令的条件，所述条件分支命令是在满足了所述条件的情况下使对所述监视对象程序的举动进行监视的处理开始的命令。

监视装置和监视方法

技术领域

[0001] 本发明涉及监视装置和监视方法。

背景技术

[0002] 以往，在计算机系统上，为了对来自攻击者的非法侵入或具有恶意的程序运行的情况进行检测而提出一种从系统的举动对可疑的动作进行检测的主机型侵入检测方法(例如参照非专利文献1)。

[0003] 作为对系统上的应用程序的动作进行监视的方法，执行一种对API(Application Programming Interface:应用程序接口)调用进行监视的方法。该API是将各种系统调用抽象化的函数。例如，应用程序能够通过API以不直接意识硬件的方式就简单地进行文件输入输出或者通信控制等。通过对这种API调用进行监视，能够获取API的种类或者输入了何种参数等的日志信息，其结果能够监视应用程序的一系列的动作。

[0004] 此外，作为获取这种日志信息的方法可以使用API钩子(hook)，该API钩子在API调用或者API的执行过程中使控制流程迁移而获取日志信息。例如，作为通过API钩子获取日志信息的方法，公知有如下的方法：在各API的开头命令中插入跳转命令或者调用命令使处理流程迁移到别的命令串进行日志信息的获取，再将处理流程返回到原来的API。

[0005] 现有技术文献

[0006] 非专利文献1：“Detours:Binary Interception of Win32Functions”、3rd USENIX Windows(注册商标)NT Symposium、USENIX、Galen Hunt著、July 1999

发明内容

[0007] 发明要解决的课题

[0008] 但是，在上述的以往的技术中，因为为了使用API钩子而在开头命令中插入跳转命令或者调用命令，所以恶性程序不进行本来的恶性动作，从而存在无法适当地监视API调用的情况。

[0009] 例如，在使用上述的API钩子的方法中，因为通过在API的开头命令中插入跳转命令或者调用命令而使处理流程迁移到别的命令串，所以恶性程序通过确认API的开头命令而能够判别API是否被钩挂。因此，恶性程序在判断出API被钩挂的情况下不进行本来的恶性动作，从而存在无法适当地监视API调用的可能性。

[0010] 因此，该发明是为了解决上述的以往的技术课题而完成的，其目的在于以恶性程序侧无法得知对API调用进行监视的方式适当地对API调用进行监视。

[0011] 用于解决课题的手段

[0012] 为了解决上述的课题并实现目的，监视装置的特征在于，具有：判定部，其对所执行的程序是否是监视对象程序进行判定；以及命令赋予部，其在由所述判定部判定为所执行的程序是监视对象程序的情况下，在每次加载安装有由该监视对象程序调用的钩挂对象API的DLL时，动态地生成第1命令串和条件分支命令，并在包含于所述钩挂对象API中的命

令串之前按顺序赋予所述第1命令串和所述条件分支命令，所述第1命令串始终满足执行条件分支命令的条件，所述条件分支命令是在满足了所述条件的情况下使对所述监视对象程序的举动进行监视的处理开始的命令。

[0013] 此外，监视方法的特征在于，包含：判定步骤，对所执行的程序是否是监视对象程序进行判定；以及命令赋予步骤，在由所述判定步骤判定为所执行的程序是监视对象程序的情况下，在每次加载安装有由该监视对象程序所调用的钩挂对象API的DLL时，动态地生成第1命令串和条件分支命令，并在包含于所述钩挂对象API中的命令串之前按顺序赋予所述第1命令串和所述条件分支命令，所述第1命令串始终满足执行条件分支命令的条件，所述条件分支命令是在满足了所述条件的情况下使对所述监视对象程序的举动进行监视的处理开始的命令。

[0014] 发明效果

[0015] 本申请中公开的监视装置和监视方法能够以恶性程序侧无法得知对API调用进行监视的方式对API调用进行适当地监视。

附图说明

[0016] 图1是示出第一实施方式涉及的监视系统的结构的框图。

[0017] 图2是示出监视部的结构的功能框图。

[0018] 图3是说明正常时的API的执行顺序的图。

[0019] 图4是说明正常时的API调用的图。

[0020] 图5是对API钩子进行说明的图。

[0021] 图6是说明将API的开头命令置换成条件分支命令以及始终满足条件分支命令的条件的命令串的情况下的处理的图。

[0022] 图7是说明API钩挂时的API的执行顺序的图。

[0023] 图8是示出第一实施方式涉及的监视部的处理动作的流程图。

[0024] 图9是示出执行监视程序的计算机的图。

具体实施方式

[0025] 下面参照附图对该发明涉及的监视装置和监视方法的优选实施方式进行详细地说明。另外，本发明不限于该实施方式。

[0026] (第1实施方式)

[0027] 首先使用图1对第一实施方式涉及的监视系统的概要进行说明。图1是示出本实施例涉及的监视系统的结构的框图。如该图所示，本实施例涉及的监视系统具有监视部10、监视对象程序20、监视对象外程序20a以及OS(Operation System：操作系统)30。此外，在此说明的结构仅是一例，监视系统也可以以其它的各种方式实施。

[0028] 监视部10是在OS 30上运行的程序，从监视对象程序20中监视与主机资源或者进程相关的访问。例如，该监视部10被作为由多个模块构成的模块库来安装。各监视部10以进程单位(程序单位)或者模块单位对监视对象程序20的动作进行监视。

[0029] 监视对象程序20是成为监视系统监视的对象的程序，与监视部10一起运行。例如，该监视对象程序20是计算机病毒或蠕虫等恶性程序、或者是可能具有脆弱性的无法信赖的

程序等。

[0030] 在第一实施方式涉及的监视系统中，被监视对象程序20调用的API(Application Programming Interface:应用程序接口)的开头命令被置换成钩挂用命令串，该钩挂用命令串中被赋予了条件分支命令以及在条件分支命令之前始终满足该条件分支命令的命令串。

[0031] 另外，监视对象外程序20a是成为监视系统监视的对象外的程序。此外，假设监视部10事先掌握成为监视对象程序20的恶性程序的特征，能够判别监视对象程序20与监视对象外程序20a。

[0032] OS 30是用于使各种程序运行的软件，向监视对象程序20或者监视对象外程序20a提供通过系统调用而调用的各种功能。

[0033] 在第一实施方式涉及的监视系统的监视部10中，在判定为执行监视对象程序20的情况下，向监视对象程序20插入对钩挂对象API进行钩挂的“hook_API”。然后，如果该监视对象程序20执行被钩挂的API，则经由钩子API执行系统调用，所以监视部10能够对所有的监视对象程序20的API执行进行监视。

[0034] 即，进行由监视对象程序20执行所钩挂的API并经由钩子API对监视对象程序20的举动进行监视的处理(例如，获取API的输入值等日志信息的处理)的结果是，能够对监视对象程序20的API执行进行监视。

[0035] 接着，使用图2对监视部10的结构进行说明。图2是示出监视部10的结构的功能框图。如图2所示，监视部10特别地具有程序执行部11、判定部12、DLL(Dynamic Link Library:动态链接库)监视部13、DLL读入控制部14以及命令赋予部15。此外，在此对各功能部所具有的功能的概要进行说明，之后对各功能部所执行的处理进行详细地说明。

[0036] 程序执行部11使监视对象程序20或者监视对象外程序20a运行。首先使用图3对执行监视对象外程序20a的情况下的正常的处理进行说明。图3是说明正常时的API的执行顺序的图。如图3所例示的那样，在监视对象外程序20a使用API_1A的情况下，程序向正在运行的进程的存储器中读入安装有API_1A的DLL_1。然后，监视对象外程序20a执行在DLL_1中定义的API_1A。

[0037] 此外，程序执行部11在使监视对象程序20运行的情况下，向监视对象程序20插入记述有用于调用“hook_API”的API的代码之后，执行监视对象程序20。具体地说，程序执行部11在使监视对象程序20运行的情况下，向监视对象程序20插入记述有用于调用“hook_API”的API的代码，然后在监视系统所提供的半透过性的虚拟隔离环境中执行监视对象程序20。

[0038] 这样，程序执行部11通过向监视对象程序20插入记述有用于调用“hook_API”的API的代码之后执行监视对象程序20，从而进行对监视对象程序20的举动进行监视的处理(例如获取API的输入值等日志信息的处理)，对监视对象程序20的API执行进行监视。

[0039] 判定部12对执行的程序是否是监视对象程序进行判定。具体地说，判定部12对被程序执行部11所执行的程序是监视对象程序20还是监视对象外程序20a进行判定，并将判定的结果通知给DLL监视部13。

[0040] DLL监视部13对安装有钩挂对象API的DLL是否被加载进行监视。具体地说，DLL监视部13如果从判定部12接收到监视对象程序20被执行的判定结果，则DLL监视部13监视安

装有钩挂对象API的DLL是否被读入到监视对象程序20运行的进程的存储器中。然后，DLL监视部13在安装有钩挂对象API的DLL被读入到存储器中的情况下，将被读入的情况通知给DLL读入控制部14。

[0041] DLL读入控制部14将钩挂DLL加载到对象程序的进程存储器中。具体地说，DLL读入控制部14在从DLL监视部13接收到安装有钩挂对象API的DLL被读入到存储器的通知的情况下，使安装有钩子API的DLL读入到对象进程的存储器中。

[0042] 命令赋予部15在监视对象程序20被执行时，在包含于被该监视对象程序20所调用的API中的命令串之前按顺序赋予满足规定的条件的命令串、以及作为在满足了规定的条件的情况下使规定的控制处理开始的命令的条件分支命令。

[0043] 具体地说，命令赋予部15在由DLL读入控制部14将安装有钩子API的DLL读入到对象进程的存储器中的情况下，作为DLL的初始化处理，命令赋予部15将钩挂对象API的开头周边命令替换成始终满足规定的条件的算术运算或者比较等命令串、以及在满足了规定的条件下使流程迁移到执行对监视对象程序20的举动进行监视的处理(例如获取日志信息等的处理)的控制流程的条件分支命令。

[0044] 例如，也可以预先设定多个将始终满足规定的条件的命令串、以及在满足了规定的条件的情况下使流程迁移到获取日志信息等的控制流程的条件分支命令对应起来的组，命令赋予部15从多组中随机获取一组，并在由监视对象程序20所调用的API开头命令中赋予在所获取组中相互对应的条件分支命令和命令串。此外，命令赋予部15也可以生成始终满足规定的条件的命令串、以及在满足了规定的条件的情况下使流程迁移到获取日志信息等的控制流程的条件分支命令。

[0045] 在此，使用图4对正常的API调用进行说明。图4是说明正常时的API调用的图。如图4中例示的那样，如果进行监视对象程序所包含的API的调用，则执行API的内部处理。

[0046] 然后，使用通过置换这样的API的开头命令使处理迁移到别的命令串并进行日志获取等的基于API钩子的API调用监视方法。例如，如图5所示，通过在API的开头命令中赋予钩子函数的调用命令，而迁移到获取API的输入值等日志的处理流程，之后处理流程返回到原来的API。

[0047] 在此，以往作为钩子函数的调用命令，API的开头命令被置换成跳转命令或者调用命令。与此相对，在本实施方式涉及的监视部10中，作为钩子函数的调用命令，将API的开头命令置换成条件分支命令以及始终满足条件分支命令的算术运算或比较等的命令串。

[0048] 例如，假设条件分支命令是根据标志寄存器的状态来决定是否使控制流程迁移到指定的地址的命令。在此，所谓的标志寄存器是作为算术运算或比较等命令的结果而存储“0”或“1”的寄存器。此外，上述的命令串是满足条件分支命令的条件的算术运算或比较等的命令串。假设如果根据该命令串执行算术运算或比较的处理，则标志寄存器的状态成为始终满足条件分支命令的条件的状态。

[0049] 即，通过将API的开头命令置换成条件分支命令以及始终满足该条件分支命令的条件的算术运算或比较等的命令串，能够生成与控制流程一定迁移到指定的地址的跳转命令同义的命令串。该条件分支命令与始终满足该条件分支命令的命令串的组合可以生成无数的组合，可以成为从程序侧无法推测的命令串，能够难以判别是否被钩挂。

[0050] 即，监视部10通过将控制流程的迁移用赋予条件分支命令以及始终满足条件分支

命令的条件的命令串来代替跳转命令或调用命令，能够使API的调用源的程序判断出API未被钩挂，从而能够促使执行API，所以能够使恶性程序侧无从得知对API调用进行监视的情况，从而能够适当地对API调用进行监视。

[0051] 而且，也可以在钩挂API时每次动态地生成命令串。因此，从API调用源的程序中无法通过比较API的开头与特定的命令串的图案来判断是否被钩挂。这样能够以不向恶性程序暴露的方式进行API钩挂，从而对程序的动作进行监视。

[0052] 在此，使用图6的例子对将API的开头命令置换成条件分支命令以及始终满足条件分支命令的条件的命令串的情况进行说明。图6是对将API的开头命令置换成条件分支命令以及始终满足条件分支命令的条件的命令串的情况的处理进行说明的图。如图6所例示的那样，将被监视对象程序20所调用的API_A的开头命令置换成标志寄存器调整用代码和条件分支命令。在此，所谓的标志寄存器调整用编码是进行算术运算或比较等的命令串，是当通过该命令串执行算术运算或比较的处理时调整标志寄存器的状态使其成为满足条件分支命令的条件的状态的命令代码。

[0053] 此外，所谓的条件分支命令是在标志寄存器的状态满足规定的条件的情况下使控制流程迁移到指定的地址的命令。如上述所述，因为被赋予到条件分支命令之前的标志寄存器调整用代码调整标志寄存器的状态使其成为满足规定的条件的状态，所以能够使控制流程迁移到指定的地址。该控制流程例如是进行获取API_A的输入值等日志的处理的控制流程。

[0054] 此外，因为将API的开头置换成条件分支命令以及始终满足条件分支命令的条件的命令串，所以在控制流程返回到原来的API之前执行完被覆盖的原程序的处理，之后，控制流程返回到原来的API，执行未被覆盖的原程序处理即API的内部处理。这样，能够无矛盾地执行API。

[0055] 下面使用图7说明API的执行顺序。如果监视对象程序20(在图7中记载为“程序”)被执行，则该监视对象程序20读入安装有API_1A的DLL_1(参照图7的“1”)。然后，监视对象程序20在加载完DLL_1之后读入安装有通过钩挂API_1A来进行处理的hook_API_1A的DLL_1_hook(参照图7的“2”)。

[0056] 在此，如果DLL_1_hook被监视对象程序20读入，则监视部10在执行了API_1A的情况下在API_1A的开头命令串中赋予条件分支命令以及在条件分支命令之前赋予始终满足条件分支命令的条件的命令串，以使命令迁移到hook_API_1A。然后，如果监视对象程序20执行在DLL_1中所定义的API_1A，则在执行完钩挂处理之后，执行API(参照图7的“3”)。

[0057] 这样，监视部10在DLL_1_hook被监视对象程序20读入时，在执行了API_1A的情况下，在API_1A的开头命令串中赋予条件分支命令以及始终满足该条件分支命令的命令串，以使命令迁移到hook_API_1A。此外，虽然也可以考虑事先替换DLL_1的文件本身，但是因为必须要修改源代码进行编译，所以需要公开安装有钩挂对象API的DLL的源代码。此外，虽然技术上也可以直接替换二进制本身，但是需要二进制的逆向工程，根据程序不同还有些被禁止，有不能够替换的情况。

[0058] (监视部的处理)

[0059] 下面使用图8对第一实施方式涉及的监视部10的处理进行说明。图8是示出第一实施方式涉及的监视部的处理动作的流程图。

[0060] 如图8所示，监视部10的判定部12判定监视对象程序是否被执行(步骤S101)，在判定为监视对象程序已被执行的情况下(步骤S101肯定)，判定安装有钩挂对象API的DLL是否被加载(步骤S102)。

[0061] 其结果，在监视部10的DLL监视部13判定为安装有钩挂对象API的DLL未被加载的情况下(步骤S102否定)，返回到步骤S102的处理。此外，在判定为安装有钩挂对象API的DLL被加载的情况下(步骤S102肯定)，监视部10的DLL读入控制部14将钩挂DLL加载到对象程序的进程存储器(步骤S103)。

[0062] 然后，监视部10的命令赋予部15对钩挂对象API进行置换，以将开头命令迁移到钩子API(步骤S104)。具体地说，如果安装有钩子API的DLL被DLL读入控制部14读入到对象进程的存储器中，则命令赋予部15作为DLL的初始化处理针对钩挂对象API将开头周边命令置换成条件分支命令以及始终满足条件分支命令的条件的算术运算或比较等的命令串。

[0063] 然后，监视部10判定所有的钩挂对象API是否被钩挂(步骤S105)。即，因为钩挂对象API有时被安装于多个DLL，或者DLL的读入时机不限于程序的起动时，所以直到钩挂对象API全部能够钩挂为止都需要对监视对象程序进行监视。

[0064] 其结果，监视部10在判定为未钩挂所有的钩挂对象API的情况下(步骤S105否定)，返回到步骤S102的处理。此外，监视部10在判定为已钩挂所有的钩挂对象API的情况下(步骤S105肯定)，使处理结束。

[0065] (第一实施方式的效果)

[0066] 如上述所述，第一实施方式涉及的监视部10对执行的程序是否是监视对象程序进行判定。然后，监视部10在判定为执行的程序是监视对象程序的情况下，在包含于被该监视对象程序所调用的函数中的命令串之前按顺序赋予满足规定的条件的命令串、以及作为在满足了规定的条件的情况下使规定的控制处理开始的命令的条件分支命令。

[0067] 因此，因为第一实施方式涉及的监视部10能够生成无数用于使控制流程迁移的命令串、以及条件分支命令的图案，所以从监视对象程序20侧难以判别API被钩挂的情况，所以能够在不妨碍程序的运行监视的情况下对监视对象程序20的动作进行监视。这样，在监视部10中能够以恶性程序侧无法得知对API调用进行监视的方式对API调用进行适当地监视。

[0068] 此外，第一实施方式涉及的监视部10从将始终满足规定的条件的命令串、以及在满足了规定的条件的情况下使流程迁移到获取日志信息等的控制流程的条件分支命令对应起来的多组中随机获取一组，并在被监视对象程序20所调用的API开头命令中赋予在所获取的组中相互对应的条件分支命令和命令串。因此，因为能够事先生成多个用于使控制流程迁移的命令串和条件分支命令的组合，所以容易在API的开头命令中赋予满足规定的条件的命令串、以及作为在满足了规定的条件的情况下使规定的控制处理开始的命令的条件分支命令。此外，因为是随机获取，所以能够难以从监视对象程序20侧判别API被钩挂的情况。

[0069] 此外，第一实施方式涉及的监视部10按顺序赋予进行运算或者比较以使存储于标志寄存器的值满足规定的条件的第1命令串、以及作为在标志寄存器满足了规定的条件的情况下使规定的控制处理开始的命令的条件分支命令。因此，可以使用既有的标志寄存器生成无数用于使控制流程迁移的命令串、以及条件分支命令的图案。

[0070] 此外，第一实施方式涉及的监视部10执行包含条件分支命令和命令串的API，并执行对监视对象程序20的举动进行监视的处理作为控制处理。例如，监视部10通过获取API的输入值等日志而能够对监视对象程序20的举动进行监视，并能够适当地对监视对象程序的动作进行监视。

[0071] 此外，在进行核心层的系统调用监视的情况下，不仅捕获监视对象程序还捕获也包含无关的程序在内的OS上的所有的系统调用事件，需要从中通过进程ID等判别是否是监视对象程序来进行处理。因此，核心层的系统调用监视产生用于对无关的系统调用事件进行处理的开销。另一方面，第一实施方式涉及的监视部10因为只对监视对象程序20进行监视，所以不产生上述的开销。

[0072] (第二实施方式)

[0073] 在此之前对本发明的实施例进行了说明，但是本发明也可以在上述的实施方式以外以各种不同的方式实施。因此，下面作为第二实施方式对包含于本发明的其它的实施方式进行说明。

[0074] (1)系统结构等

[0075] 此外，图示的各装置的各结构要素是功能概念的内容，在物理上未必需要如图示那样构成。即，各装置的分散/合并的具体方式并不限于图示的内容，能够根据各种负荷或使用状况等使其全部或者一部分以任意的单位在功能上或者物理上进行分散/合并地构成。例如，也可以合并判定部12与命令赋予部15。而且，各装置所进行的各处理功能其全部或者任意的一部分可以通过CPU和在该CPU中解析执行的程序实现、或者能够实现为基于布线逻辑的硬件。

[0076] 此外，在本实施例所说明的各处理当中，作为自动执行的内容而被说明的处理的全部或者一部分可以手动执行、或者作为手动执行的内容而被说明的全部或者一部分可以通过公知的方法而自动执行。另外，对于包含在上述文件中或附图中所示出的处理顺序、控制顺序、具体的名称以及各种数据或参数在内的信息，除了特殊说明的情况以外能够任意变更。

[0077] (2)程序

[0078] 此外，也可以将在上述实施方式中说明的监视部10执行的处理制成以计算机能够执行的语言记述的程序。例如，也可以将监视部10执行的处理制成以计算机能够执行的语言记述的监视程序。在这种情况下，通过计算机执行监视程序也能够获得与上述实施方式相同的效果。而且，也可以将所涉及的监视程序记录到计算机能够读取的记录介质中，通过使计算机读入并执行被记录到该记录介质的监视程序而实现与上述第一实施方式相同的处理。下面对执行与在图2中示出的监视部10同样的功能的监视程序的计算机的一例进行说明。

[0079] 图9是示出执行监视程序的计算机1000的图。如图9所例示的那样，计算机1000例如具有存储器1010、CPU 1020、硬盘驱动器接口1030、磁盘驱动器接口1040、串行端口接口1050、视频适配器1060以及网络接口1070，这些各部由总线1080连接。

[0080] 如图9所例示的那样，存储器1010包含ROM(Read Only Memory:只读存储器)1011和RAM 1012。ROM 1011例如存储BIOS(Basic Input Output System:基本输入输出系统)等引导程序。如图9所例示的那样，硬盘驱动器接口1030与硬盘驱动器1031连接。如图9所例示

的那样，磁盘驱动器接口1040与磁盘驱动器1041连接。例如磁盘或光盘等可以装卸的存储介质插入到磁盘驱动器中。如图9所例示的那样，串行端口接口1050例如与鼠标1051、键盘1052连接。如图9所例示的那样，视频适配器1060例如与显示器1061连接。

[0081] 在此，如图9所例示的那样，硬盘驱动器1031例如存储OS 1091、应用程序1092、程序模块1093以及程序数据1094。即，上述的监视程序作为记述有被计算机1000执行的指令的程序模块而存储于例如硬件驱动器1031中。

[0082] 此外，在上述实施方式中说明的各种数据作为程序数据而存储于例如存储器1010或硬盘驱动器1031中。而且，CPU 1020根据需要将存储于存储器1010或硬盘驱动器1031中的程序模块1093或者程序数据1094读出到RAM 1012，并执行访问监视顺序、访问控制顺序、进程监视顺序以及进程控制顺序。

[0083] 此外，监视程序涉及的程序模块1093或程序数据1094并不限于存储于硬盘驱动器1031的情况，例如也可以存储于可以装卸的存储介质并经由磁盘驱动器等被CPU 1020读出。或者，监视程序涉及的程序模块1093或程序数据1094也可以存储于经由网络(LAN(Local Area Network:局域网)、WAN(Wide Area Network:广域网)等)连接的其它的计算机中，并经由网络接口1070被CPU 1020读出。

[0084] 符号说明

[0085] 10:监视部;11:程序执行部;12:判定部;13:DLL监视部;14:DLL读入控制部;15:命令赋予部;20:监视对象程序;20a:监视对象外程序;30:OS。

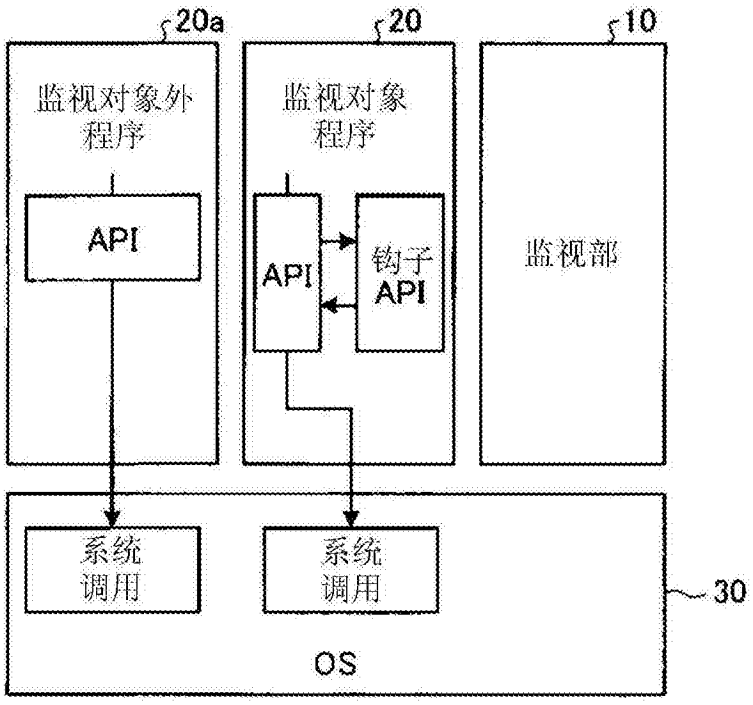

20a
20
10
监视对象外程序
监视对象程序
API
API
钩子API
监视部
系统调用
系统调用
30
OS

图1

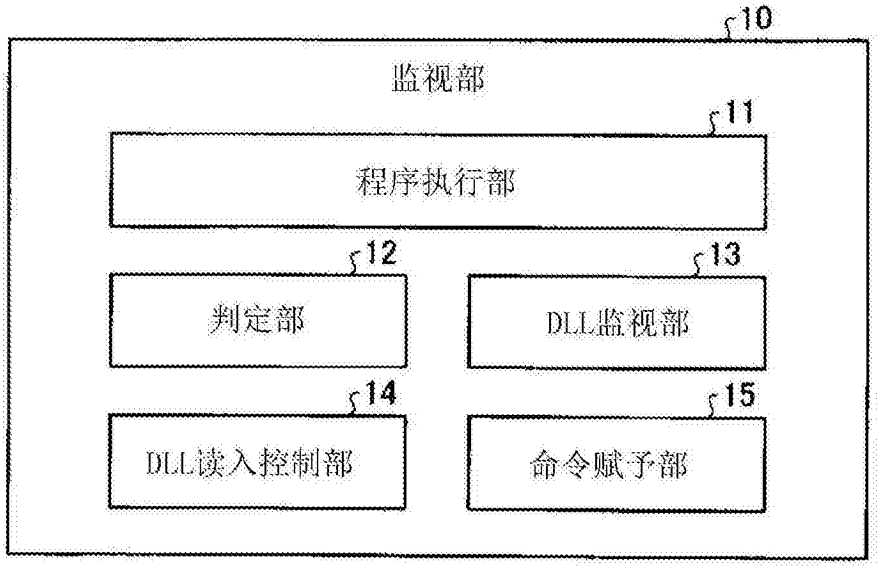

10
监视部
11
程序执行部
12
判定部
13
DLL监视部
14
DLL读入控制部
15
命令赋予部

图2

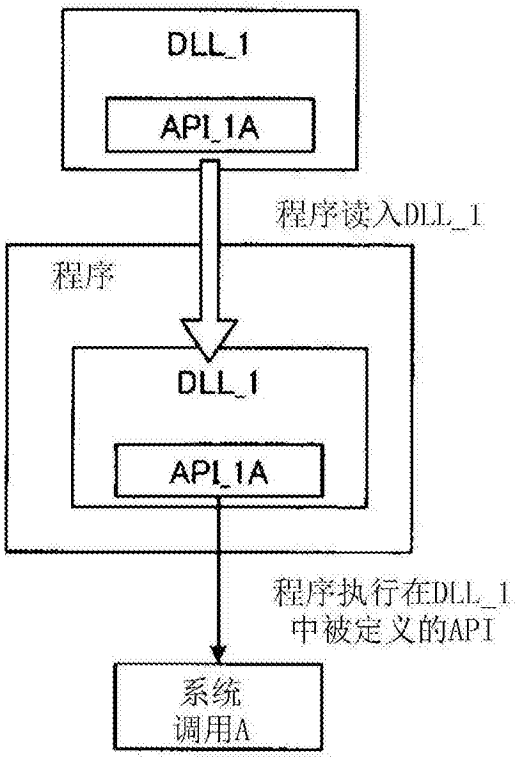

DLL_1
API_1A
程序读入DLL_1
程序
DLL_1
API_1A
程序执行在DLL_1
中被定义的API
系统
调用A

图3

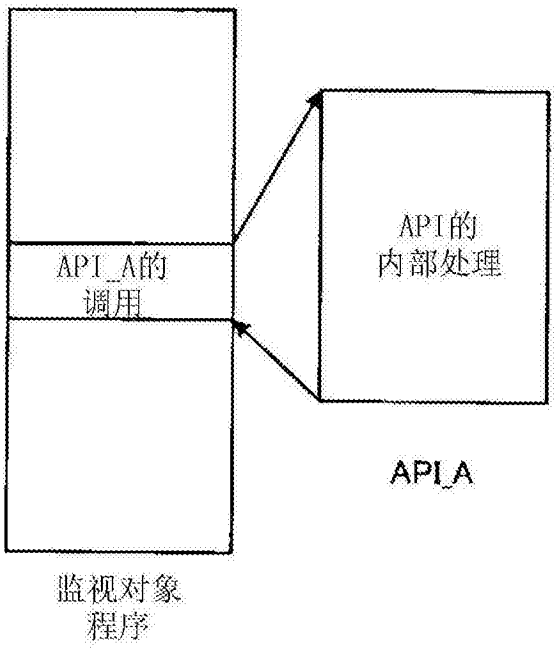

API_A的
调用
API的
内部处理
API_A
监视对象
程序

图4

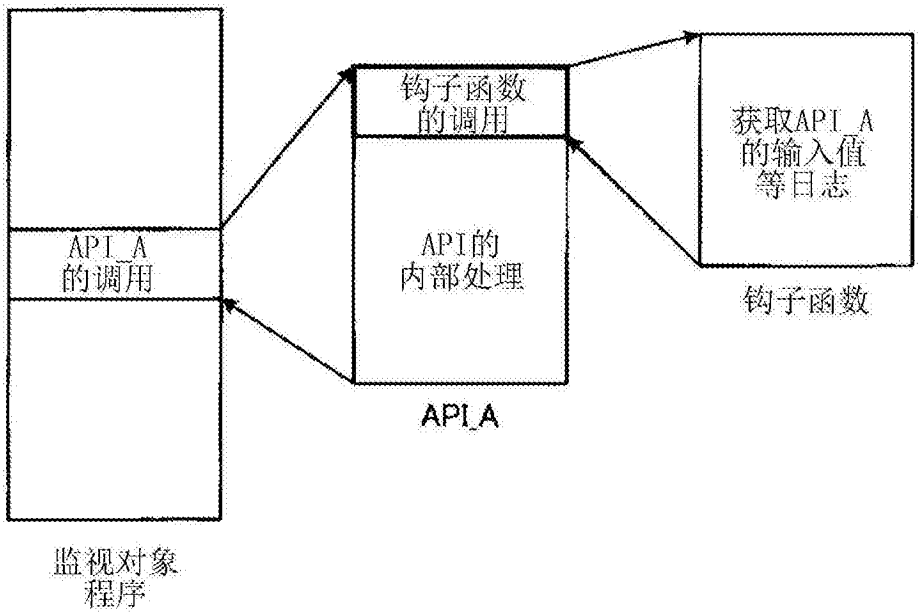

钩子函数
的调用
获取API_A
的输入值
等日志
API_A
的调用
API的
内部处理
钩子函数
API_A
监视对象
程序

图5

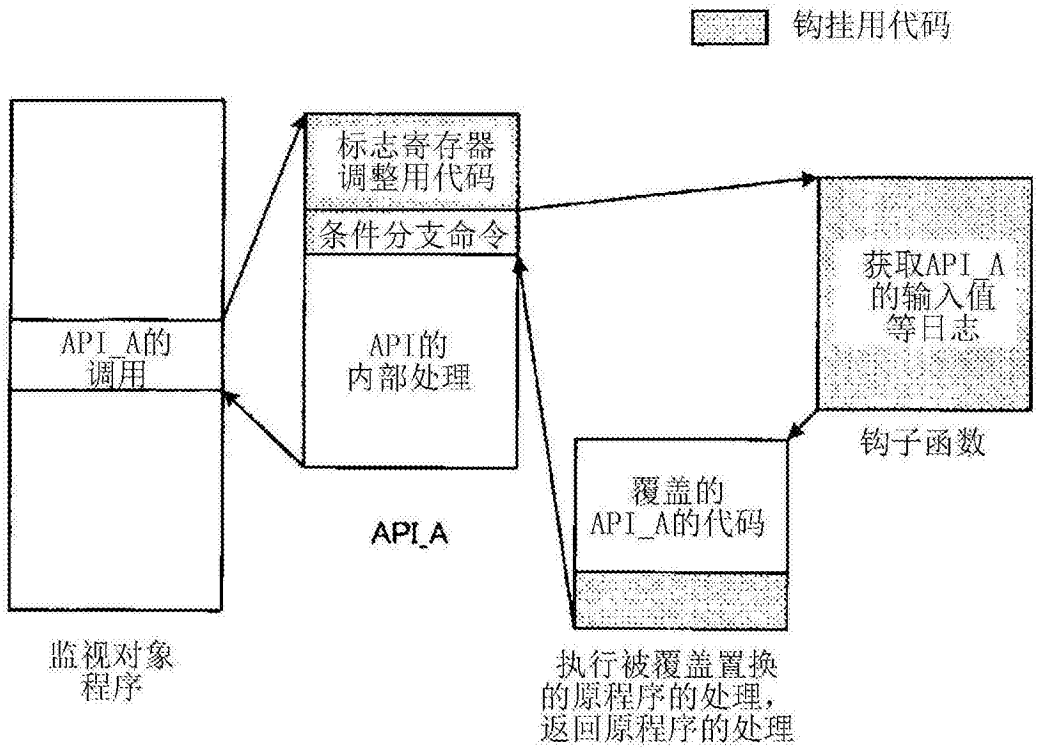

钩挂用代码
标志寄存器
调整用代码
条件分支命令
获取API_A
的输入值
等日志
API_A的
调用
API的
内部处理
钩子函数
覆盖的
API_A的代码
API_A
监视对象
程序
执行被覆盖置换
的原程序的处理，
返回原程序的处理

图6

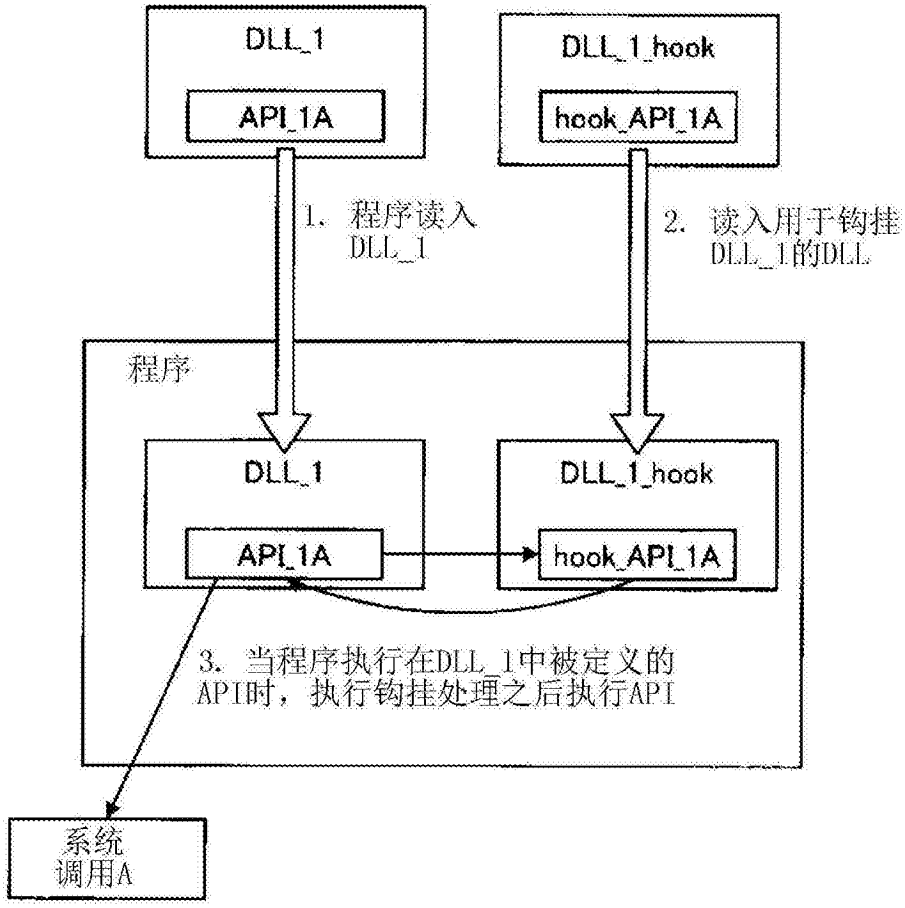

DLL_1
API_1A
DLL_1_hook
hook_API_1A
1. 程序读入
DLL_1
2. 读入用于钩挂
DLL_1的DLL
程序
DLL_1
API_1A
DLL_1_hook
hook_API_1A
3. 当程序执行在DLL_1中被定义的
API时，执行钩挂处理之后执行API
系统
调用A

图7

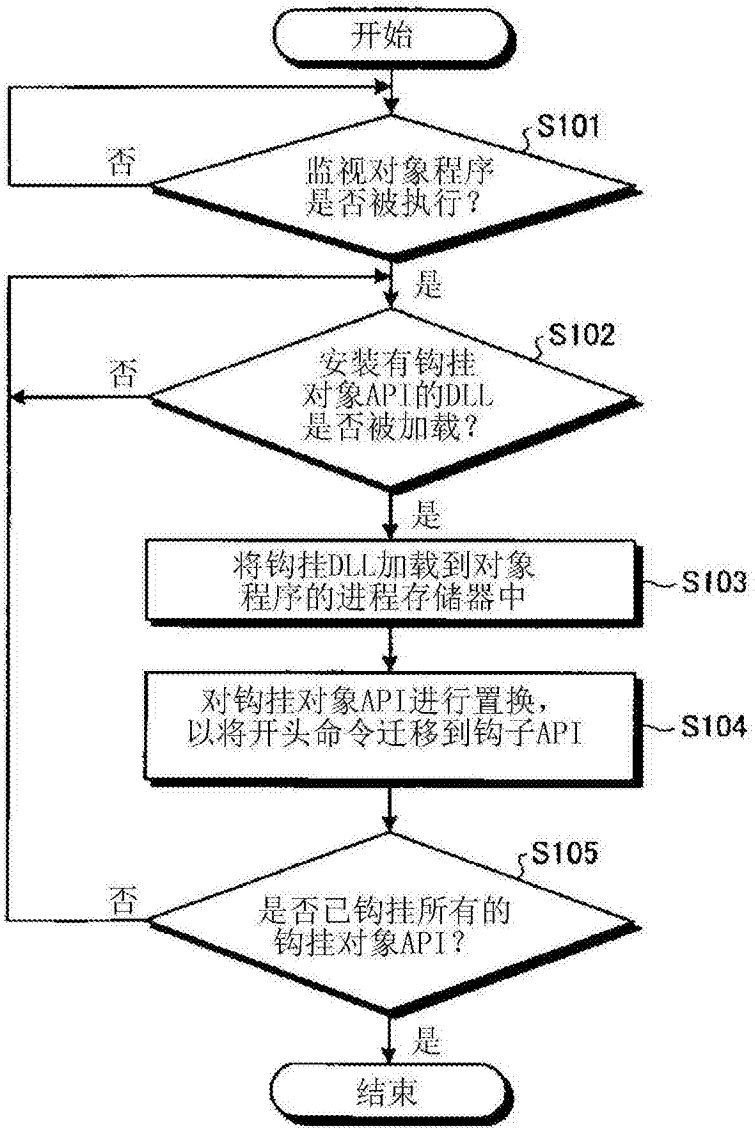

开始
S101
否
监视对象程序
是否被执行？
是
S102
否
安装有钩挂
对象API的DLL
是否被加载？
是
将钩挂DLL加载到对象
程序的进程存储器中
S103
对钩挂对象API进行置换，
以将开头命令迁移到钩子API
S104
S105
否
是否已钩挂所有的
钩挂对象API？
是
结束

图8

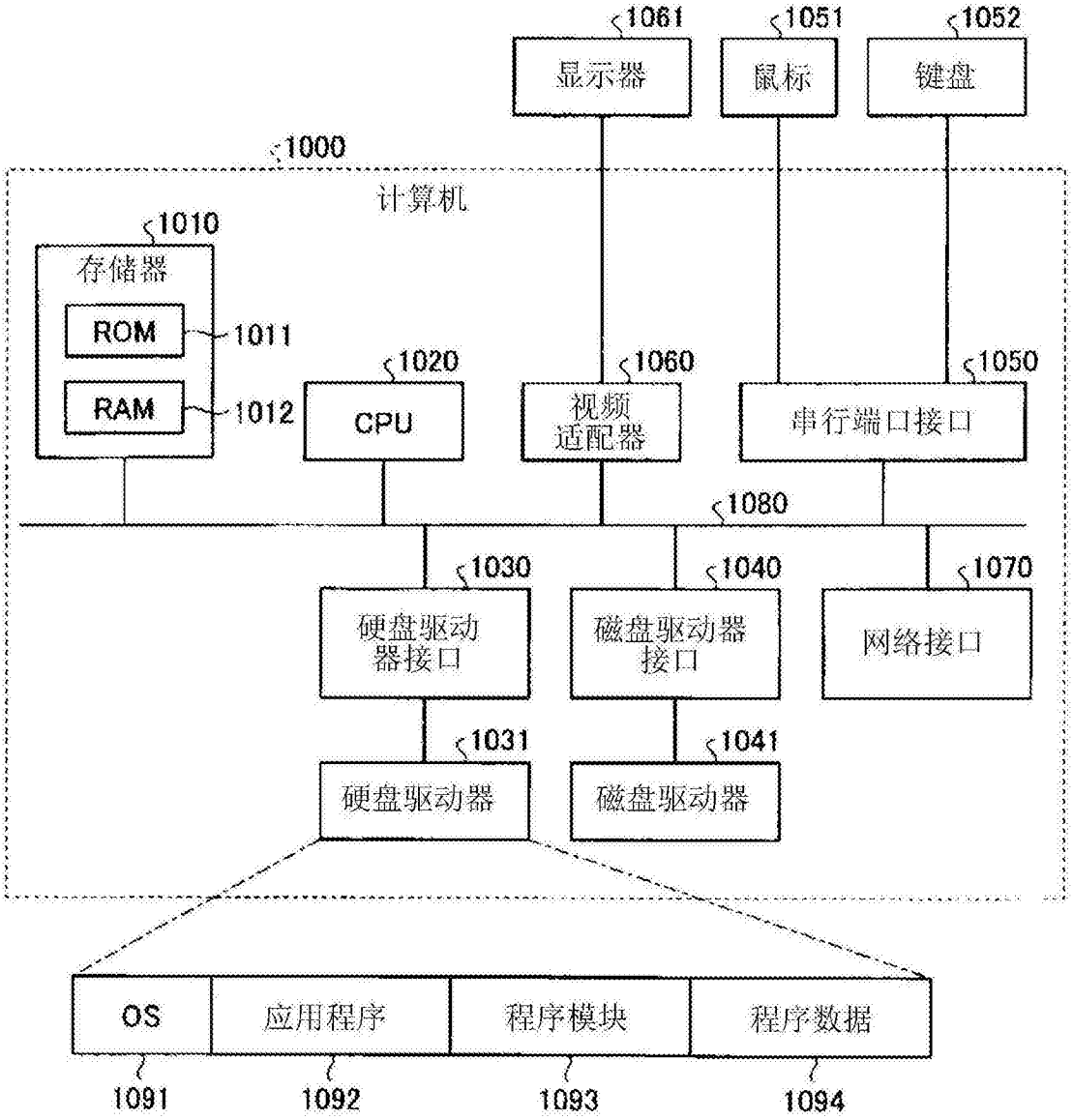

1061
显示器
1051
鼠标
1052
键盘
1000
计算机
1010
存储器
ROM
1011
RAM
1012
1020
CPU
1060
视频
适配器
1050
串行端口接口
1080
1030
硬盘驱动
器接口
1040
磁盘驱动器
接口
1070
网络接口
1031
硬盘驱动器
1041
磁盘驱动器
OS
应用程序
程序模块
程序数据
1091
1092
1093
1094

图9