工学硕士学位论文

开放式数控组件软件跨平台问题研究

吴连祥

北京工业大学

2005年5月

国内图书分类号：

单位代码： 10005
学号：S200207003
密级： 秘密

北京工业大学工学硕士论文

题 目 开放式数控组件软件跨平台问题研究

英文并列
题 目 THE RESEARCH OF CROSS-PLATFORM ABOUT THE OPEN NUMERICAL COMPONENT SOFTWARES

研究生姓名： 吴 连 祥

专 业： 计算机软件与理论 研究方向： 软件工程

导师姓名： 刘 燕 军 **职 称：** 副研究员

论文报告提交日期 2005 年 5 月 学位授予日期

授予单位名称和地址 北京工业大学 北京市朝阳区平乐园 100 号 邮编 10022

中文摘要

开放式数控系统是当前数控技术发展的主要趋势，它的目标是要允许用户根据需要选配功能模块并集成，以适应不同应用系统的需求。开放式控制系统有三个不同的开放程度，环境的开放，接口的开放，组件模块的开放，三者体现了不同的实现技术。组件的可二进制重用特性及语言无关性，使得开发的数控应用组件实现二进制兼容，并提高开放式控制系统的可扩展性，且有助于将系统的接口标准化，从而在很大程度上实现了数控系统的开放性问题。

然而这种开放性还是受特定系统平台制约的，开放式数控系统要真正实现完全意义上的开放性，还应是与具体操作系统平台无关的。从而可以使数控界开发人员专注于领域产品设计开发，不被具体操作系统环境束缚。而且数控产品要提升自己的竞争力，减少或避免重复开发，对操作系统平台依赖的问题也需要得到解决。然而，基于一个系统平台上开发的应用，最终都必然地与这个系统的平台模块有着紧密的依赖关系，因此，跨平的实质在一定程度上是如何解决这种依赖关系的问题。

本文以科泰公司的和欣 2.0 系统和微软公司的 Windows CE.NET 4.2 系统这两款嵌入式操作系统来作为对象，根据跨平台的实现需要，文中分别详细分析了两平台组件机制实现原理，对两系统的平台服务模块进行了细致的分析，并讨论了两操作系统及其可执行模块的一些特点。然后以和欣2.0到Windows CE的跨平台为主，依据两平台的特点，制定出跨平台实现方案。方案以在一个系统上仿真另一个系统的系统平台模块的方式来解决应用对于具体平台依赖问题，从而做到一种嵌入式系统下的应用软件在另一个嵌入式系统上做到跨平台二进制兼容运行。为很好地满足开放式数控系统的开放性要求，组件技术日益受到重视，因此组件软件的跨平台必须解决。二进制跨平台二进制级跨平台为寻求数控软件更大的开放目标具有重要意义。

关键字：跨平台，平台仿真，组件技术，和欣平台，平台开放性

Abstract

The main trend of numerical control technology is the open numerical control system, to adapt itself to requirements of different application system, an open numerical control system allows a user to select function modules and integrate them according to requirement. There are three open grades about open control system, they are the open of environment, the open of interface and the open of component, which incarnate different implement technology. With the binary code reusability and the program language independence of a component, component makes numerical control application binary code compatible, improves the expansibility of open numerical control system, and it is also beneficial to the standardization of system interface, hence, component implements the open problem of numerical control system largely.

However, the openness benefit from component is confined by special system platform, the entire open of an open numerical control system should be irrespective of a given OS platform. The developer of application, thereby, can be liberated from a special OS and absorbed in the domain product. Furthermore, to improve the competition of a numerical control product and reduce or avoid repeat development, the platform relied problem should be resolved. Application based on a OS platform must be rely on modules of the OS platform closely, the essential of cross-platform, therefore, is to solve such dependency.

The Elastos 2.0 OS of Koretide and Windows CE.NET 4.2 of Microsoft are introduced in this article. According to the requirement of cross-platform implement, it analyzes, respectively, principle about component realization in detail, platform service modules of two OS are

also be analyzed detailedly, moreover, it make a expatiation about the two OS and characteristics of executable module. Finally, it gives the implement scheme about cross-platform according to the peculiarity of two platform, which considers mostly how to run applications based on Elastos 2.0 on Windows CE OS. It solve this problem by emulating platform service modules of one system on another system, so that application runs one system platform can run on another system in binary code. In order to satify the requirement of open numerical control system commendably, importance is attached increasely to component technology, so it should be resolved about component software cross-platform. It will show a grate importance for larger extent openness for numerical control applications when the cross-platform problem is implement.

Keywords cross-platform, platform emulating, component technology, Elastos platform, openness of platform

目录

第 1 章 绪论

数控装置的开放趋势是20世纪90年代制造设备自动化领域的热点之一。数控系统“开放”的要求来自于生产方式的发展，也来自于用户和机床厂商对附加技术及成本的要求，制造信息的集成化和生产系统的分期化也促进了控制器的开放化。随着控制器的PC化和控制方案的软件化，以PC机为基础的开放式数控系统呈蓬勃发展之势[3]。平台的开放性也必将是开放式数控系统的又一个最重要需求。

1.1　课题背景及实际意义

我国是一个机床生产与应用大国，然而在数控技术领域与世界水平却有较大的差距，传统数控系统的绝大部分市场都被西门子，FANUC和A-B等大公司所垄断，国产数控系统在国内数控机床上所占份额很小，特别在高档数控系统方面，国产系统在国内的市场份额还不到2%[4]。随着高性能、低价格PC机硬件的普及，基于软件的控制器技术和伺服技术得以充分发展。

而当新计算机的体系结构、指令系统或操作系统发生变化时，相当一部分数控应用软件将无法正常运行。显然，放弃原有成熟的数控应用系统而重新设计研制和投资购置基于不同机器环境或操作系统环境的，具有相似甚至相同功能的数控应用软件，需要大量的“二次投资”，这是不明智也是不现实的，必将造成企业较大的资金浪费和重复开支[15]。甚至不同数控企业，所做事情相同，只因为所基于的操作系统不同，而使开发的数控软件组件不能共享，这其实也是一种意义上的“重复建设”。那么，如何解决上述问题，首先人们考虑的是软件移植，即通过对一个软件在新的运行环境做一些必要的技术处理，或者只需对此软件做少量修改、加工，就能由一种机器环境或操作系统环境搬到另一种机器环境或操作系统环境下运行。

软件移植是对这种问题的一种行之有效的解决办法，但却是不彻底的解决办法，而且，移植工作繁重，对于大型应用软件而言移植质量很可能难以完全得到保障。如果能使一个数控应用系统，以二进制形式在另一个系统上运行起来，则能够实现完全意义上的软件共享，并且不用担心由于软件移植而可能会出现的

软件质量问题。开放式控制体系的提出，意味着普通控制器开发商可以在价廉物美的通用计算机上开发出具有独立功能特点，但又符合国际标准规范，可运行于各种软、硬件平台上的数控系统[3]。中国人才资源丰富，我们应抓住当今发达国家推广开放式数控系统的契机，从系统平台层面，谋求数控软件资源的最大共享，从而充分利用中国人才资源优势，使他们专注于数控领域设计，谋求数控应用的更大开放性，并提高数控应用的质量。

1.2 课题来源及主要研究内容

本课题是北京市科委项目“数字化制造装备基础部件”中的子课题“开放式数控软件系统重构技术的研究”。目标是以和欣2.0操作系统及Windows CE操作系统为研究对象,对两操作系统的主要系统服务模块进行分析，应用中间件原理，在Windows CE操作系统上对和欣2.0的主要系统服务模块进行仿真，即在Windows CE操作系统上,实现一个虚拟的和欣(Elastos)2.0操作系统平台,从而在和欣2.0系统上的应用软件能够以二进制形式直接在Windows CE操作系统上运行,并分析其跨平台性能，论证这种跨平台方式的现实可行性。

1.3 国内外文献综述

跨平台技术主要分为软件跨平台和硬件跨平台。硬件跨平台是指操作系统能够使得应用程序运行在不同的硬件平台上，例如 ARM 和 X86 硬件平台。软件跨平台是指应用软件能够在多种操作系统或者不同数据库平台上运行，跨操作系统平台的代表产品有 Java、.NET 及 Linux 上的 Wine 等。

在解决应用的跨平台问题上，常见的有重编译、中间件技术或虚拟机等途径[13]。重编译指将应用在新的系统下进行修改编译，使之在新的平台下能运行，即为软件开发了不同系统平台的版本，然而这种解决办法并不能很好地做到软件复用；中间件技术则是通过开发大量通用的跨平台中间件代码，使得基于这些中间件代码上开发的应用不作或少作修改就能放到不同的不兼容的平台上运行。中间件处理在操作系统、网络和数据库之上、应用软件之下，是起承上启下作用的应用支撑平台[5]。中间件是位于计算机硬件和操作系统之上支持应用软件开发和运行的系统软件[7]，其核心思想是分层，通过在应用层与底层基础软件间增

批注 [lyj1]: 从软件工程的角度这也是复用，所以不要这么说。只是这么复用，离不开软件开发人员。或者说是面向编程人员的复用

加一层，屏蔽底层复杂的技术细节，实现对底层的透明访问，为应用的开发、部署与管理提供支持；虚拟机则是通过在不同操作系统平台提供一致的虚拟机，使得基于这个虚拟机的应用实现跨平台运行。虚拟机技术的根本思想是在操作系统和程序之间提供虚拟的执行引擎，由它全面负责程序的执行。通过提供一套虚拟的中间指令集，确保了程序的跨平台性。源代码首先被编译成中间代码，然后程序执行时，由执行引擎即时编译或者解释执行[8]。在实现应用程序跨平台的技术当中，Java是源代码级兼容，而Wine则是二进制兼容。

1.4　跨平台技术发展状况

基于虚拟机思想实现跨平台有大家所熟知的 Java，.NET 及 Wine 等。

Java 虚拟机

Java虚拟机是Java平台的基石，它确保了应用程序具有操作系统和硬件的平台独立性。在某种程度上相当于一台抽象的计算机，它有一套指令集并具有不同操作的内存区域[8]。Java平台可分两部分，即Java虚拟机和Java API类库。在Java平台的结构中，见图 1-1，Java虚拟机在核心的位置。它的下方是移植接口。移植接口由两部分组成。其中依赖于平台的部分称为适配器。Java虚拟机通过移植接口在具体的平台和操作系统上实现。在Java虚拟机的上方是Java的基本类库和API。利用JavaAPI编写的应用程序(application)和小程序(Java applet)可以在任何Java平台上运行而无需考虑底层平台，从而实现了Java的平台无关性[18]。

批注 [lyj2]: 程度？

批注 [lyj3]: ？ jvm——java虚拟机?注意文字说明与图中标识的一致性。

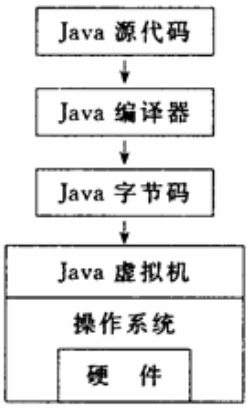


图 1-1　Java 结构示意图

Fig. 1-1 The architecture about Java

其关键技术有两点：一是在服务器端进行编译。而编译过程只是将Java源程序编译为与平台无关的中间代码（字节码）；二是在客户端安装Java虚拟机，以解释方式执行中间代码程序。

批注 [lyj4]: 其？

.NET 虚拟机

.NET 技术的基石是公共语言基础架构(Common Language Infrastructure, CLI)。CLI 为可执行代码和代码执行环境提供规范。而公共类型系统(Common Type System, CTS)处于 CLI 的核心地位，它被编译器，各种工具及 CLI 共享。它定义了如何在 CLI 中定义、使用和管理类型的规则。图 1-2 给出了 CLI 运行的基本框图。

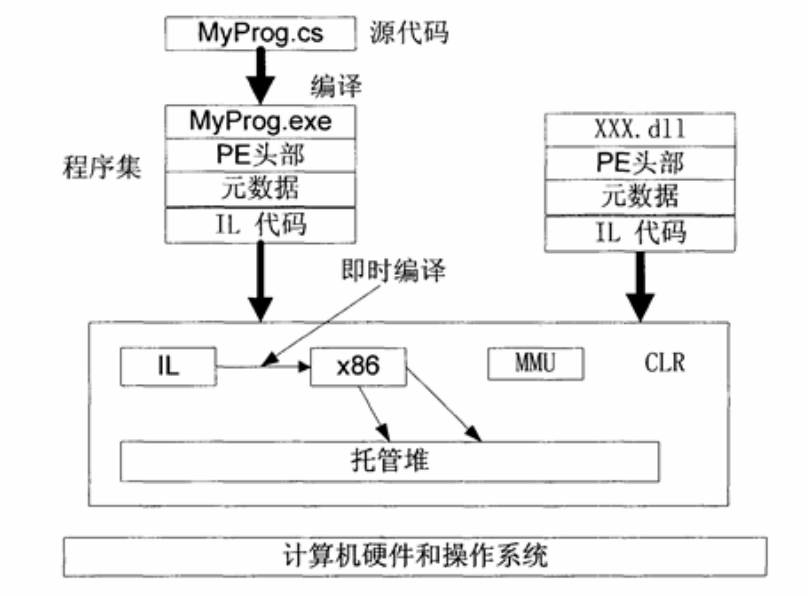


图 1-2 CLI 的执行框图

Fig. 1-2 The process about CLI

以公共类型系统作为基础的.NET 框架，实现了跨语言集成、类型安全及代码的高性能执行。CLI 主要由公共类型系统、元数据、公共语言规范、虚拟执行系统、文件格式、公共中间语言及类库等组成。

元数据描述和引用公共类型系统定义的类型，它以独立于各种编程语言的格式存储，并提供一种公共的交换机制，在编译器、调试器及代码执行环境等各部分之间共享数据。虚拟执行系统实现和确保 CTS 模型，装载和执行基于 CLI 的程

序，提供执行托管代码和数据的服务，并利用元数据在运行期间连接不同模块。公共中间语言是一种基于堆栈的语言，高级语言首先被编译成中间语言，在执行之前再被编译成机器语言或被解释运行。类库则提供可重用的组件单元。

在.NET 上，高级语言首先被编译成程序集，它包括元数据和中间代码。当程序执行时，执行引擎首先装载所需的程序集。然后中间代码被即时编译成本地代码，在本地 CPU 上执行。程序执行时，使用托管堆作为其内存分配空间，在其中分配的无用内存，将自动被内存管理单元回收。由文献[8,17-18]可知。

Wine 仿真器

Wine 是在 Linux 上建立的一个模拟 Windows API 的环境，在 Linux 系统下，应用程序运行在 Wine 之上，Wine 仿真了绝大部分 Windows 的 API 接口函数，因此应用程序可以直接运行在 Linux 系统上，不需要进行二次编译。如图 1-3 所示。

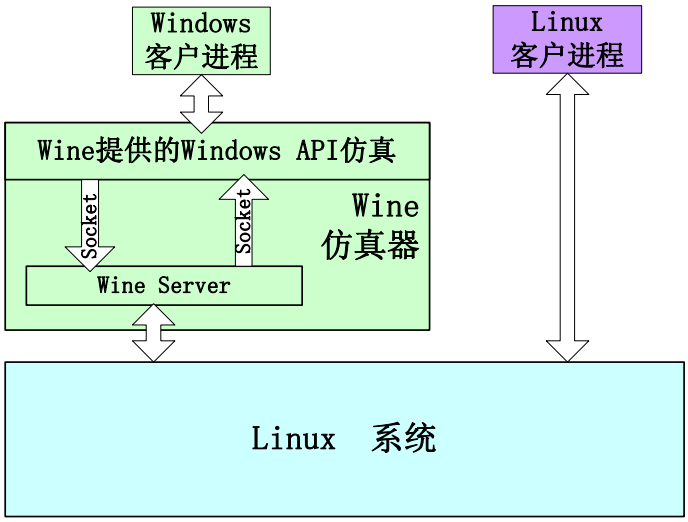


图 1-3 Linux 上的 Wine 仿真器

Fig. 1-3 The wine emulator on Linux

在实现在 Linux 上运行 Windows 应用程序的过程中，Wine 主要解决了以下几个技术难点：

1) 操作系统系统调用的转换。

2) 可执行程序从 PE-COFF 格式到 EFL 格式的转换。

Wine主要由两个模块组成：API仿真层和Wine Server。Wine Server主要实

现PE-COFF格式的应用程序的加载以及系统调用的转换。应用程序的系统调用通过API仿真层转换为系统调用请求，然后通过Socket发送该请求给Wine Server，最后由Wine Server 调用Linux内核的系统调用。系统调用成功后返回相应API仿真层。在API仿真层和Wine Server通信的过程中，采用了服务请求的机制，类似于C/S的体系结构。对于图形相关的应用程序系统调用，是通过Wine Server与 X Server 之间的系统调用来完成。Wine因为只是在Linux系统上模拟Windows应用程序的运行环境，因此在Wine上开发的应用程序是不能够运行在Windows系统上的[12]。

批注 [lyj5]: ?

虚拟机是没有实际计算机硬件与之相对应的，完全由软件实现功能的、基于特定操作系统的“纯软计算机”。虚拟机作为独立的应用运行于主操作系统之上，基于虚拟机平台，用户即可引导、运行特定的客户操作系统及应用。虚拟机的核心机制是通过软件方式为客户操作系统提供一个虚拟的硬件平台，对于客户操作系统而言，这个虚拟平台等价于一个“裸机”系统。

采用中间件思想实现跨平台，则多见于不同的数据库、异构的网络平台、及分布式应用等领域。最早具有中间件技术思想及功能的软件是IBM的CICS，但由于CICS不是分布式环境的产物，因此人们一般把Tuxedo作为第一个严格意义上的中间件产品，它是由国外最有名的中间件厂商BEA推向市场的。尽管中间件概念很早就产生，但中间件的技术的广泛运用却是在近 10 年之中。IBM的中间件MQSeries可说是个相对成熟的消息中间件产品[10]。

1.5 本章小结

本章对软件工程角度及现实需求，引出了软件跨平台问题的来源，对国内外在跨平台问题的处理方法作了一定的介绍，并简要地叙述了java虚拟机，.NET虚拟机，Wine仿真器在实现跨平台问题上的解决方法及其实现原理。

第 2 章 和欣 2.0 及其平台分析

2.1　和欣 2.0 系统简介

科泰世纪科技有限公司研发的和欣 2.0 操作系统是一款 32 位嵌入式操作系统。它具有基于微内核，具有多进程、多线程、抢占式、基于线程的多优先级任务调度等特性。并且它的体积小，速度快，适合网络时代的绝大部分嵌入式信息设备。和欣 2.0 系统提供的功能模块全部基于动态链库或 CAR（Carefree Application Run-Time）组件技术，因此是可拆卸的模块，应用系统可以按照需要剪裁组装，或在运行时动态加载必要的 CAR 组件。

2.1.1 和欣 2.0 系统体系结构

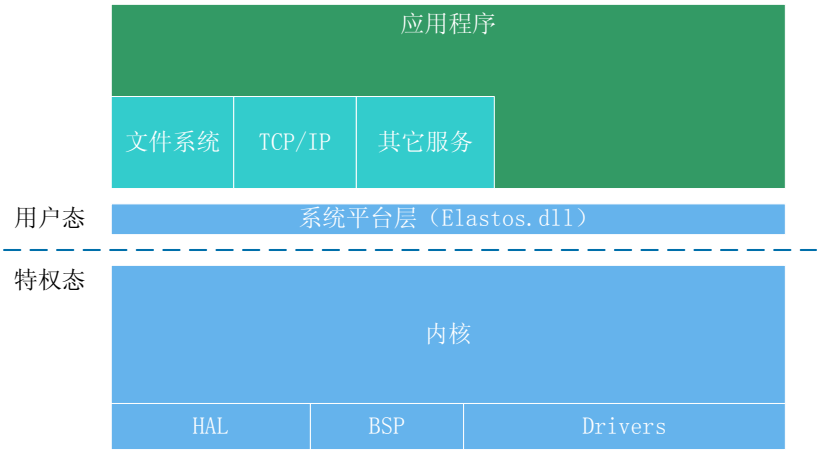


图 2-1 和欣 2.0 体系结构

Fig. 2-1 The architecture about Elastos 2.0

如图 2-1 所示，和欣 2.0 由由微内核、组件支持模块、系统服务器组成的[1]。

- 微内核：主要可分为 4 大部分：硬件抽象层（对硬件的抽象描述，为该层之上的软件模块提供统一的接口）；内存管理（规范化的内存管理接口，虚拟内存管理）；任务管理（进程管理的基本支持，支持多进程，多线程）；

进程间通信。

批注 [lyj6]: 这是否也意味着组件必须以进程为单位？

- 组件支持模块：提供了对 CAR 组件的支持，实现了组件运行环境。组件支持模块并不是独立于微内核单独存在的，微内核中的进程间通讯部分为其提供了必要的支持功能。
- 系统服务器：在微内核体系结构的操作系统中，文件系统、设备驱动、网络支持等系统服务是由系统服务器提供的。在和欣操作系统中，系统服务器都是以动态链接库的形式存在。[1]

2.1.2 和欣 2.0 系统的虚拟地址空间

和欣 2.0 系统是一个 32 位的嵌入式系统，每个进程有 4GB 的虚拟地址空间，从而使得嵌入式应用程序有很大的虚拟运行空间。但和欣的进程地址空间分为两大部分：内核空间、用户空间。0 到 2GB 为用户进程空间，2GB 到到 4GB 为内核空间。如图 2-2 所示。

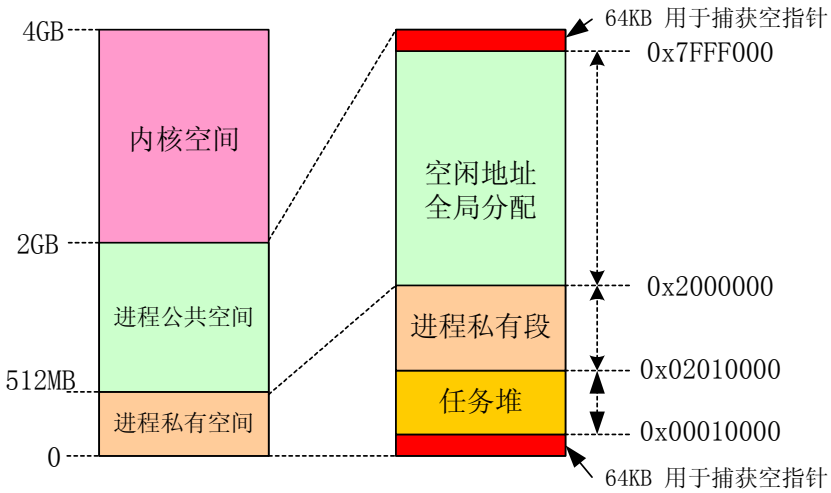


图 2-2 和欣 2.0 进程空间的分配

Fig. 2-2 The allocation about process space on Elastos 2.0

内核程序运行在内核地址空间内。用户进程不能直接访问此地址空间，所以实际上和欣 2.0 的应用程序可使用 2GB 的虚拟地址空间。两端分别有 64KB 是不能访问的，用于捕获空指针或越界指针，具体分配如下：

- 0x00000000 到 0x0000FFFF：不可访问，用于捕获空指针操作；
- 0x00010000 到 0x0200FFFF：共 32MB，作为本进程的用户堆；

•0x02010000 到 0x1FFFFFFF：线程栈和其它私有段空间，每个线程分配 1MB 地址空间；

• 0x20000000 到 0x7FFEFFFF：公共进程空间，用于装载 EXE、DLL 等模块，本空间全局分配，以便 DLL 的共享；

• 0x7FFF0000到0x7FFFFFFF：不可访问，用于捕获越界指针操作。[1]

2.2　和欣 2.0 系统的 CAR 组件机制

2.2.1　CAR 组件简介

和欣的 CAR（Component Application Run-Time）组件技术是面向组件编程的编程模型，它规定了一组组件间相互调用的标准，使得二进制组件能够自描述，能够在运行时动态链接。CAR 兼容微软的 COM，CAR 很大程度地借鉴了 COM 技术，保持了和 COM 的兼容性。但是和微软 COM 相比，CAR 删除了 COM 中过时的约定，对组件的类别、组件的自描述数据类型类型进行了扩展，禁止用户定义 COM 的非自描述接口，完备了组件及其接口的自描述功能。对 COM 的用户界面进行了简化包装，易学易用。CAR 技术就是在总结面向对象编程、面向组件编程技术的发展历史和经验，为更好地支持面向以 Web Service（WEB 服务）为代表的下一代网络应用软件开发而发明的。

为了在资源有限的嵌入式系统中实现面向中间件编程技术，同时又能得到C/C++的运行效率，CAR 没有使用 JAVA 和.NET 的基于中间代码-虚拟机的机制，而是采用了用 C++编程，用和欣 SDK 提供的工具直接生成运行于和欣组件运行平台的二进制代码的机制。用 C++编程实现组件技术，使得更多的程序员能够充分运用自己熟悉的编程语言知识和开发经验，很容易掌握面向组件、中间件编程的技术。

面向对象的C++编程语言对C语言的重要扩展有：封装性、重用性、多态性。CAR技术依然支持这三个重要扩展，但是是在针对二进制代码的级别实现的，数据和操作数据的函数封装在类中之后，可以被子类继承，从一个类的数据结构通过实例化可以得到多个具有相同数据结构的对象，C++的源代码一经编译，则在编译时生成对象方法的虚函数表，在程序运行时，运行时系统根据程序的执行情

批注 [lyj7]: ？ 态

况查找这张虚函数表，以找到具体的方法并执行，从而形成对多态性的支持。

CAR编程模型中，运行时系统对虚函数表的操作有了重大改进，CAR的运行系统可对多个虚函数表再生成一张表，这张新表中的记录除了指向各个虚函数表的指针之外，还有对这个新表自身的说明。在CAR编程模型中，这些自身说明信息称为元数据[1]。

2.2.2 CAR 组件技术优势

和欣 2.0 操作系统区别于其他商用嵌入式操作系统产品的最大优势在于其全面面向组件的技术，在操作系统层提供了对组件运行环境的支持。而且用户可以通过网络获得服务组件，并且这个组件是带有自描述信息的组件，本地系统能够为这个程序建立运行环境，自动加载运行。

这个组件化的和欣操作系统可以为我们进行嵌入式系统软件的开发带来以下好处：

- 它在嵌入式软件开发领域，导入先进的工程化软件开发技术,使得嵌入式应用的软件开发能够实现工程化、工厂化生产。从而既满足嵌入式应用领域的软件产品对嵌入式操作系统的需求，为我们从软件工程方面探索嵌入式软件的开发提供便利。
- 它可以实现组件动态加载。动态加载组件是因特网时代嵌入式系统的必要功能。新一代的软件产品不能再像以前那样由厂家将所有的功能都做好后固定在产品里，而要允许用户从网上获得自己感兴趣的组件，或者从网上下载更新的或功能更优的组件，。
- 它可以随时和动态地实现软件升级。动态加载组件的功能，同样可以用于嵌入式软件产品的升级。这样嵌入式软件开发商不必为了添加了部分功能而向用户重新发布整套软件，只需要升级个别或部分相关软件组件就行。
- 它灵活的模块化结构，便于移植和剪裁。可以很容易定制成为针对不同硬件配置的紧凑高效的嵌入式操作系统。添加或删除某些功能模块也非常简单。
- 嵌入式软件开发商容易建立自己的组件库。由于和欣系统的组件技术充分表现了组件特性，这样在不同开发阶段开发的嵌入式应用软件组件，其成

果很容易被以后的开发所共享，从而保护嵌入式软件的开发投资。其完美的软件复用能力也使得嵌入式应用软件系列产品的开发更加容易，缩短软件新产品开发周期。

和欣组件技术使得共享第三方嵌入式应用软件开发商的成果变得很容易。而向行业的组件库的建设，社会软件的丰富，就使得嵌入式设备厂家不必亲自开发所有的嵌入式应用软件，可以充分利用现有的软件组件资源，充分发挥自己的在特定领域方面的专长，从而为自己的领域产品增色[1]。

CAR 组件技术是一个实现软件工厂化生产的先进技术，可以大大提升企业的软件开发技术水平，提高软件生产效率和软件产品质量；软件工厂化生产需要有零件的标准，CAR 组件技术为建立软件标准提供了参考，有利于建立企业内、行业内的软件标准，有利于建立企业内、行业内的组件库。

2.2.3　CAR 组件元数据组织

使用元数据的必要性

在编程时，包含了一个不能直接使用的 lib，另外还需要一个接口文件（也就是.h 头文件）来描述这个 lib 里面的二进制代码。头文件让用户自己的源文件可以使用别人的二进制文件，元数据作用类似头文件。

组件以接口方式向外提供服务，则接口也需要元数据来描述才能让其他使用服务的用户使用。组件为了让接口与实现无关，从而保持了接口的不变性，使得动态升级成为可能。并且使用vptr结构将接口的内部实现隐藏起来，由接口的元数据来描述接口的函数布局和函数参数属性。接口的元数据描述的就是服务和调用之间的关系。有了这种描述，不同组件之间的调用才成为可能，组件的远程化，进程间通讯，自动生成Proxy和Stub及自动Marshalling、Unmarshalling才能正确地进行[1]。

CAR 元数据组织与实现

和欣 2.0 系统里组件的接口元数据来源于.CAR文件，该文件等同于微软的

idl文件，经过CAR编译器carc.exe产生元数据文件。一个CAR文件用来说明一个CAR组件，一个组件的描述由对接口的描述和对类的描述两个部分组成，因此一个CAR文件中就由描述这两部分的代码组成。编译的时候把元数据文件打包到组件dll中，就可以通过组件的导出函数_EzComDllGetClassObject找到。CAR文件描述了一个组件里所包含的类对象的组织信息（如类的排列顺序、包含的接口）、各个接口的信息（如接口的种类、包含的接口方法）、各个方法的信息（如接口方法参数的种类、排列顺序等）以及CLSID等信息[1]。

下面请看 dog 组件的 CAR 文件（dog.car）的内容：

```
[
    version(1.0), uuid(E217C375-747A-4961-99FA-66CB47A1FFC5),
    uunm(http://www.koretide.com/car/Dog.dll)
]
component Dog
{
    [ uuid(71D8500B-C5BD-435D-8B66-9EFD562BC39E) ]
    interface IDog {
        HRESULT WatchDoor([in] wchar_t*  pwszMaster);
    }
    [ uuid(63EBE043-F731-4104-8EE2-52407B6A2A7C) ]
    class CDog {
        interface IDog;
    }
}
```

编译 dog.car,编译器在目标文件夹生成 dog.cls 文件。然后根据生成的.cls 文件，在目标目录产生包含客户所需接口信息的头文件 dog.h 供客户端使用。还有一种自描述的方式，就是在客户文件中用 #import<dog.dll> 代替 #include<dog.h>。这是因为系统工具会在编译前使用工具 mkimport 对原文件作预处理，遇到 import 语句时，会到该组件 dll 的资源段寻找元数据生成相应的

信息，从而实现了组件 dll 自带元数据的自描述。

根据生成的 dog.cls 文件，再通过系统工具 cls2src 就可以生成文件：DogClsInfo.cpp 、 _CDog.cpp 、 _CDog.h 、 _dogpub.cpp 和 _Dog_1_0.h 。DogClsInfo.cpp 文件是提供给列/散集（marshalling）使用的一个简化版元数据；_CDog 和_CDog.h 自动生成了接口的实现，为用户省去了一项重复又易出错的工作;_dogpub.cpp 生成了_EzComDllGetClassObject 函数的实现;_Dog_1_0.h 包含了生成的 dog.h 文件的元数据。这些自动生成的文件会和用户自己编写的源文件一起生成服务器组件。

以 dog 组件为例，dog.car 文件被 car 编译器编译后产生的 DogClsInfo.cpp 如下：

```
unsigned char g_Dog_classInfo[75] = {
    0x4b, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
    0x01, 0x00, 0x00, 0x00, 0x14, 0x00, 0x00, 0x00,
    0x2a, 0x00, 0x00, 0x00, 0x43, 0xe0, 0xeb, 0x63,
    0x31, 0xf7, 0x04, 0x41, 0x8e, 0xe2, 0x52, 0x40,
    0x7b, 0x6a, 0x2a, 0x7c, 0x01, 0x00, 0x47, 0x00,
    0x00, 0x00, 0x01, 0x00, 0x40, 0x00, 0x00, 0x00,
    0x0b, 0x50, 0xd8, 0x71, 0xbd, 0xc5, 0x5d, 0x43,
    0x8b, 0x66, 0x9e, 0xfd, 0x56, 0x2b, 0xc3, 0x9e,
    0x46, 0x00, 0x00, 0x00, 0x01, 0x01, 0x52, 0x2a,
    0x00, 0x00, 0x00,
};
CIClassInfo *g_pCIClassInfo_ = (CIClassInfo *)g_Dog_classInfo;
```

对于上面 g_Dog_classInfo 所包含信息的理解可通过下面这些数据结构定义获知：

```
typedef struct __PACKED__ _CIClassInfo {
    int                     totalSize;
    int                     classNum;
    int                     interfaceNum;
```

```
CIClassEntry            *classDir;
CIInterfaceEntry        *interfaceDir;
}   CIClassInfo;

typedef struct __PACKED__ _CIClassEntry {
    CLSID                   clsid;      //16字节
    UINT16                  interfaceNum;
    CIInterfaceEntry        **interfaces;
}   CIClassEntry;

typedef struct __PACKED__ _CIInterfaceEntry {
    UINT16                  methodNumMinus3; //不含接口三标准方法
CIMethodEntry          *methods;
IID                     iid;
}   CIInterfaceEntry;

typedef struct __PACKED__ _CIMethodEntry {
    CIBaseType              *params;
    UINT8                   paramNum;
    UINT8                   reserved1;
}   CIMethodEntry;

typedef UINT8                   CIBaseType;
```

CIBaseType 各位用于描述接口方法的特性信息，如方法参数的[in]、[out]属性等。

在生成的_dogpub.cpp 文件中可通过调用 E_zComDllGetClassObject 获得相关的接口信息。_EzComDllGetClassObject 函数的定义如下：

```
STDAPI _EzComDllGetClassObject(REFCLSID clsid, REFIID riid, void** ppv)
{
```

```
    if (clsid == CLSID_ClassInfo) {
        *ppv = (void *)g_pCIClassInfo_;
        return S_OK;
    }
    if (clsid == (REFCLSID)CLSID_CDog) {
        return _g_CDogCF.QueryInterface(riid, ppv);
    }
    return CLASS_E_CLASSNOTAVAILABLE;
}
```

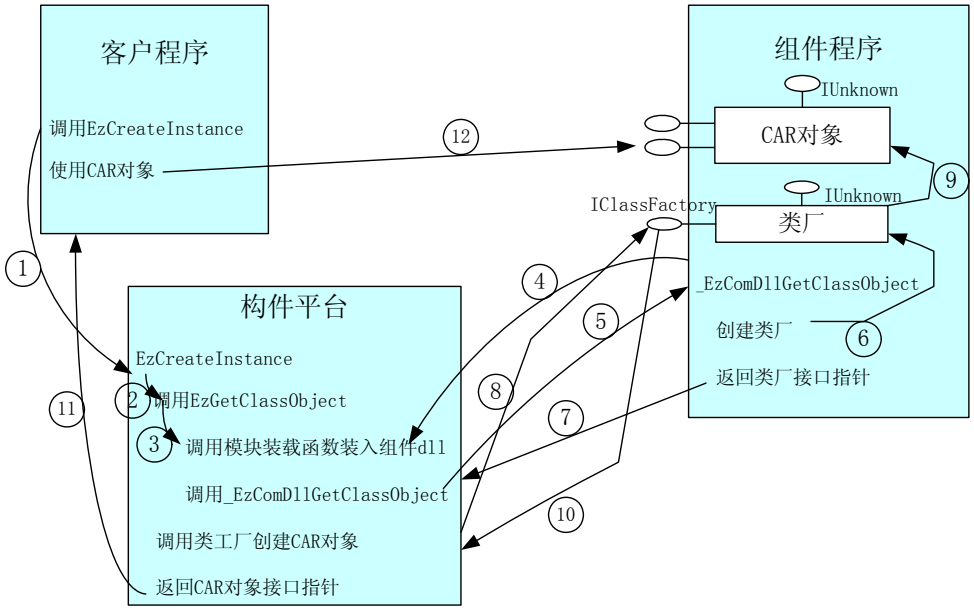
客户可通过调用_EzComDllGetClassObject 取得该组件 dll 所包含的接口元数据。

2.2.4　CAR 组件的实现

几乎所有进程内组件dll都提供一个类厂，用于向外部提供组件对象。类厂其实是组件dll内隐藏着的一个特殊组件对象(生产组件的对象)，是组件程序里面所能提供的所有对象的“生产厂”。客户程序所用到的组件对象都是由类厂生产并提供出来。和欣系统通过调用进程内组件dll的_EzComDllGetClassObject接口函数来向外提供类厂。

一般地，当和欣客户程序要创建一个和欣CAR对象时，须要调用组件运行平台的EzCreateInstance,并将相应要创建的CAR对象的clsid和iid作为参数传入，从而得到所要对象的接口。诚如上面所说，CAR对象也要通过类厂来生产的，因此，对于EzCreateInstance组件创建工作而言，它必须先获取组件类厂，这个类厂的获取又是通过调用组件运行平台的EzGetClassObject来实现，其次是调用类厂来生产出客户程序所需的CAR对象。因此，EzGetClassObject要负责在客户进程空间找到组件dll或将组件dll加载到客户的进程空间，然后调用组件dll提供的引出函数_EzComDllGetClassObject获取组件类工厂。图 2-3 给出了进程内CAR组件创建的大致过程。

批注 [lyj8]: ?



① 客户调用组件运行平台的EzCreateInstance以获取对象接口
② EzCreateInstance调用EzGetClassObject以获取类工厂接口指针
③ EzGetClassObject调用模块装载函数装入组件程序
④ 组件程序返回_EzComDllGetClassObject函数入口指针
⑤ EzGetClassObject调用_EzComDllGetClassObject以获取类厂接口指针
⑥ _EzComDllGetClassObject创建类厂
⑦ _EzComDllGetClassObject返回类工厂接口指针
⑧ EzCreateInstance调用类厂接口以创建CAR组件对象
⑨ 类厂创建CAR组件对象
⑩ 类厂返回CAR组件对象接口指针
⑪ 组件运行平台返回组件对象接口指针给客户
⑫ 客户使用CAR组件对象

图2-3 和欣上进程内组件创建过程

Fig. 3-4 The creation of in-process componet on Elastos

2.2.5 CAR 组件命名服务机制

命名服务机制

命名服务是一种以字符串为标识的服务。服务程序可以通过和欣系统平台的 API 函数 EzRegisterService 向操作系统注册自己的服务接口，而服务的使用者（即客户端程序）则可以通过 API 函数 EzFindService 来获取指定的服务接口。命名服务通过简单友好的系统 API 函数，为系统服务以及用户组件提供了一套完整的 CAR 组件使用流程，具有良好的扩展性，并支持基于组件的动态更新和升级。

CAR是科泰世纪公司拥有自主知识产权的组件技术。命名服务机制属于CAR组件技术的一部分，隶属于CAR组件技术的用户接口部分。CAR组件技术可通过命名服务机制提供一种发布，获取，使用CAR组件的方法[1]。

命名服务机制的流程

命名服务包含两个部分：服务器端和客户端。在服务器端，用户获得某个组件接口指针（可以是用户自己的组件，也可以是用户远程获得的组件指针）后，通过 EzRegisterService 函数向内核注册一个命名服务，用户选择合适的时间注销命名服务并释放资源；在客户端，其它用户可以通过 EzFindService API 函数获得相应的组件服务并应用。命名服务机制的工作流程如图 2-4。

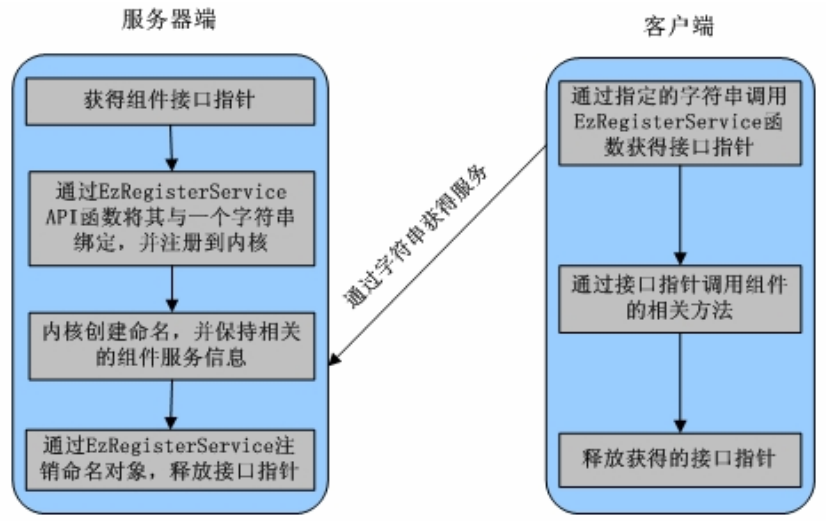


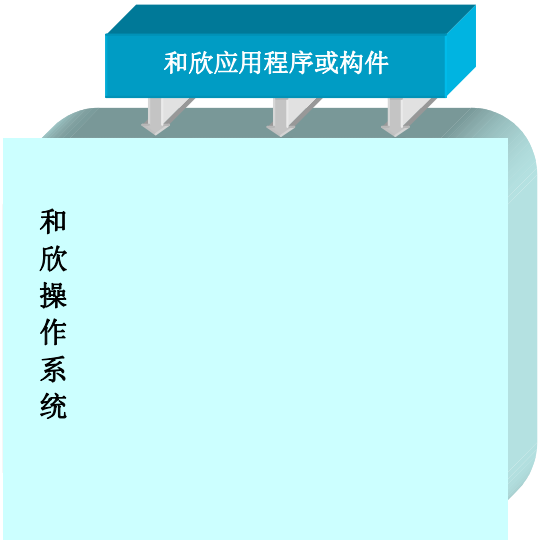
图 2-4 命名服务机制的工作流程
Fig. 2-4 The work flow about named service

命名服务完成了将服务与字符串绑定的功能，用户可以通过相应的字符串获得该服务组件对象指针[1]。

2.3 和欣 2.0 系统平台分析

2.3.1 应用程序与和欣 2.0 系统

和欣组件运行平台既为一般和欣应用程序提供运行环境，也为和欣 CAR 组件提供了运行环境。而从编程者的角度看，和欣组件运行平台提供了一套系统服务组件及系统 API，这些是在该平台上开发应用程序的基础。和欣操作系统提供的其他组件库也是通过这些系统服务组件及系统 API 实现的。



[1]

图 2-5 和欣应用程序与组件平台关系

Fig. 2-5 The relationfhip between Elastos application and the component platform

和欣应用程序及应用组件都运行于和欣组件平台上，见图 2-5。应用程序可以通过组件运行平台请求系统服务，对于用户组件，运行平台能根据二进制组件的自描述信息自动生成组件的运行环境，动态加载组件，提供组件之间的自动通信机制，也提供组件间跨进程甚至跨网络通信服务，并监控组件的运行状态，进行错误报告等。此外，还提供可干预组件运行状态的机制，如负载均衡、线程同步、访问顺序控制、安全（容错）性控制、软件使用权的控制等，以对组件的生命周期进行管理，如进程延续控制、事务元控制等。总之，组件运行平台为 CAR 组件提供了对程序员完全透明的运行环境，用户不必关心组件所在的地址空间、环境或组件所在的网络主机。它自动为组件运行提供支持，配置必要的网络协议、针对不同的输入输出设备的协议。而且，和欣系统中的进程、线程及各种

同步机制都是以组件对象的形式提供出来。

这与 JAVA 和.NET 运行机理不同，和欣组件运行平台对二进制组件直接运行提供支持，而 JAVA 和.NET 则是通过虚拟机在运行程序时解释执行中间代码。

2.3.2　和欣系统平台的构成及其关系

平台主要构成

和欣操作系统不仅为用户提供了一个组件运行平台，还提供了图形系统、设备驱动、文件系统及网络系统等的组件库系统，这些组件库通常是开发嵌入式应用系统时不可缺少的。用户开发的应用程序组件都是通过组件平台的系统接口与内核交互。这里将 2.0 系统向用户提供的主要服务模块称为系统平台模块。因此，从用户编程角度看，和欣操作系统在操作系统平台层向客户应用提供的服务模块主要由以下几大主要模块构成：

- **系统接口及组件服务支撑模块：**它向应用程序提供进程、线程对象的创建服务，允许应用程序通过及对进行、线程等对象进行操作。应用程序运行过程中的同步机制需求，组件的创建及其运行环境的开辟，组件命名服务的运作，均须通过此这模块来实现。这一个服务模块是用户可感受到的 2.0 系统核心模块，这一服务功能主要由动态库elastos.dll来提供。
- **C运行时服务模块(C run-time)**：这是一个动态运行时库，它是标准C运行库的一个有效子集，其包含了C程序运行的最基本和最常用的函数及初始化代码。如标准输入输出函数，常用字符串操作函数，内存操作及文件操作等标准C函数。和欣 2.0 系统这一服务模块主要表现为动态库elacrt.dll。
- **文件系统服务模块**：“和欣”系统是一个基于中间件技术的操作系统，该模块是一个组件化的文件服务模块，它以组件化的方式为和欣 2.0 系统提供必要的文件和目录操作。用户可根据自己需要，开发相应的组件化文件系统来替换它。和欣 2.0 系统这一服务的组件库为 vfs.dll。
- **网络服务模块**：网络应用程序不能直接与 TCP/IP 底层函数打交道。Elastos 提供一套与 UNIX-socket 和 winsock 兼容的编程接口用于进行网络编程，即为 elasock，它以套接口作为编程接口。此外，系统也提供了组件化的 socket

批注 [lyj9]: 删

批注 [lyj10]: ？

接口及 WebService 服务功能。和欣 2.0 系统网络服务由 tcpip.dll、usnet.dll、elasock.dll 等动态库及组件库实现。

- **图形系统服务模块**：“和欣”2.0 的 Atlas 图形系统是一套基于消息机制的图形系统，向用户提供常用的图形控件，在以动态链接库的形式存在于和欣 2.0 系统中。和欣 2.0 系统图形系统服务 ctrlrc.dll、drawctrl.dll、elactl32.dll、elagdi32.dll、elausr32.dll 等动态库及组件库提供。

主要模块依赖关系

前面对和欣2.0的系统平台主要构成模块进行了分析，而在这些构成模块中，elastos.dll和elacrt.dll是应用程序运行必不可少的两个支撑模块。而文件系统服务模块，网络服务模块及图形系统服务模块，则是为用户开发方便，或是为了使系统提供一个相对更完善的服务，而增加的服务模块，用户可以开发出相应的服务模块来替换它们。在和欣2.0系统上，这些模块之间存在一定的依存关系的，通过分析各平台模块中函数引入引出关系，得到它们之间的依赖关系如图2-6所示。从中可以看出，elacrt.dll及elastos.dll两平台模块的重要作用。

批注 [lyj11]: ?

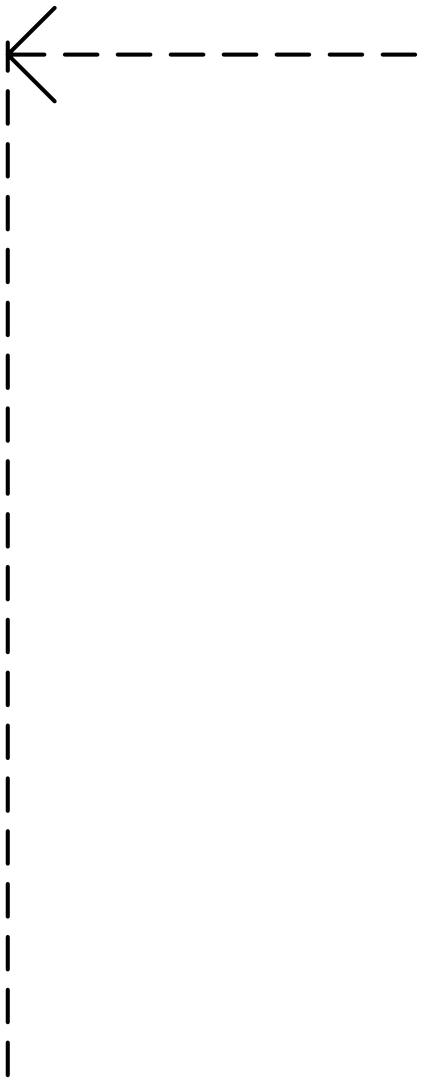


图 2-6 和欣系统平台主要模块依赖关系

Fig. 2-6 The relationship between Elastos system platform modules

2.4 本章小结

本章对和欣2.0嵌入式操作系统的体系结构及其特点作了简要介绍，并根据本文跨平台问题解决的需要，对和欣的CAR组件机制及其实现原理进行了比较详细的叙述，并最后从编程角度，在和欣2.0操作系统的平台服务层面对和欣操作系统的平台有关模块的构成其它们之间的依赖关系进行了分析。

第 3 章 Windows CE 及其平台分析

3.1 Windows CE 系统简介

这里所介绍的Windows CE系统是Windows CE.NET 4.2版本，它能为各种嵌入式系统和产品设计的一种压缩的、具有高效的、可升级的操作系统,面向从最基本的系统到高级的32位嵌入式系统开发。其多线程、多任务、全优先的操作系统环境是专门针对资源有限而设计的,这种模块化设计使嵌入式系统开发者和应用开发者能够定做各种产品。对COM(Component Object Module)组件的运行支持,是Windows CE这个嵌入式操作系统的一个重要特点。

Windows CE系统的进程有自己受保护的32MB的虚拟地址空间，进程没有优先级。它最多支持32个进程，系统启动时至少创建4个进程：内核进程、文件系统进程、GUI支持进程及加载维护系统设备驱动程序进程。一般剩余20几个进程空间可供用户创建进程用。线程是操作系统调度和运行的基本单位，Windows CE实现抢占式、基于优先级的线程调度策略。[44]

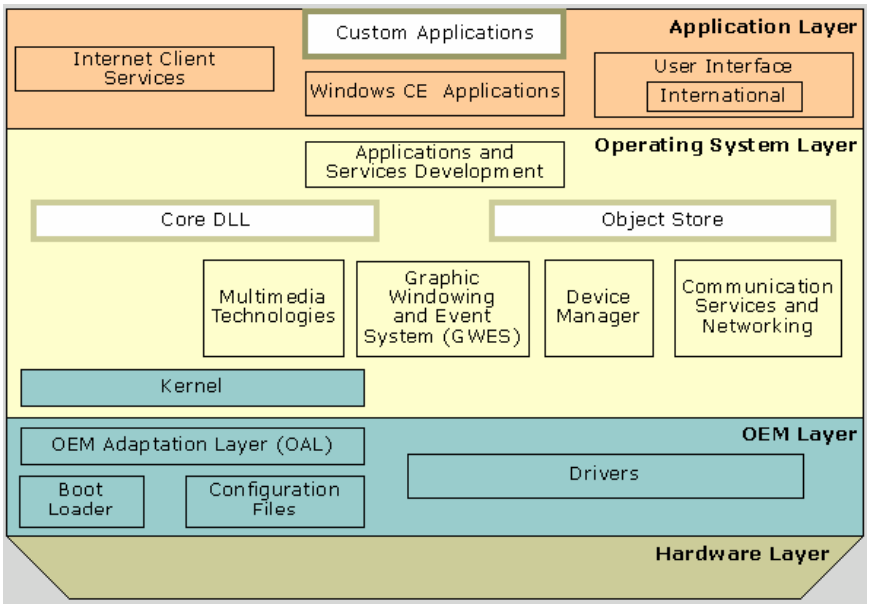
3.1.1 Windows CE 体系结构

任何一个操作系统对上要向应用程序提供服务，对下则要与各种硬件打道。Windows CE支持多种外设和网络系统，包括键盘、鼠标、触摸屏、串行口、以太网卡、调制解调器、USB设备、单频设备、并行口、打印机和存储设备。但微软的这款嵌入式操作系统并没有将所有工作都自己完成。

批注 [lyj12]: ?

批注 [lyj13]: ?

对于Windows CE而言，如图 3-1 所示，原始设备制造商（OEM）能开发硬件适配层OAL（OEM Adaptation Layer），其他开发者可以开发内部驱动程序、可安装驱动程序及运行在嵌入式设备上的应用程序。独立软件提供商ISV(Independent Software Vendor)能独立于硬件平台开发应用软件。OAL层介于Windows CE内核与硬件设备之间，实现操作系统与硬件间的通信。在操作系统层，微软负责提供内核模块、对象存储/文件系统模块、网络与通信模块及GWES等功能模块。[2]



[2]

图 3-1 Windows CE 体系结构

Fig. 3-1 The architecture of Windows CE

3.1.2 Windows CE 的地址空间

当Windows CE操作系统初化时，系统会创建一个4G的虚拟地址空间，这个地址空间为系统中的所有进程所共享，如图3-2所示。但是在这个4G的虚拟地址空间的低端，分出33个32MB大小的槽(slot)，每个槽分配给当前系统中运行的一个进程，因此Windows CE最多支持32个进程，且每个实际的应用程序进程可有的虚拟地址空间为32MB。Window CE中0号槽则用于分配给当前活动的进程，且0号槽的最低64KB的块供系统使用。当Windows CE进行进程切换时，须要重新映射地址空间，将旧进程移出0号槽，并将新进程装入0号槽，0号槽空

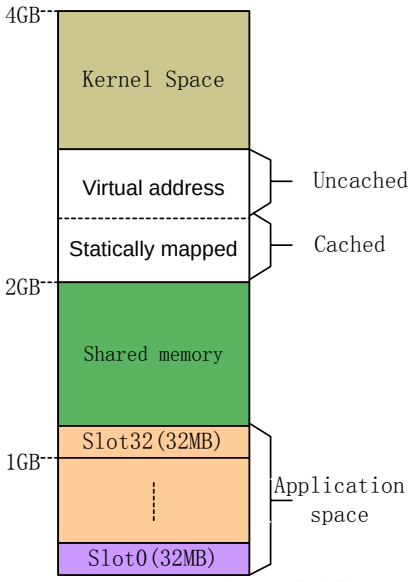


图3-2 Windows CE虚拟内存结构

Fig. 3-2 The virtual memory architecture of Windows CE

间结构如图3-3所示。33号槽以上的地址空间被用于内存映像文件和操作系统。当一个Windows CE的程序启动时，操作系统要为之创建一个进程，并选择一个可用的空闲槽给新建进程，同时为之创建进程堆和线程栈，且将进程所用到的动态链接库加载进来，但这些栈或堆及动态库所用到的虚拟内存都在同一个32MB的槽内[44]。

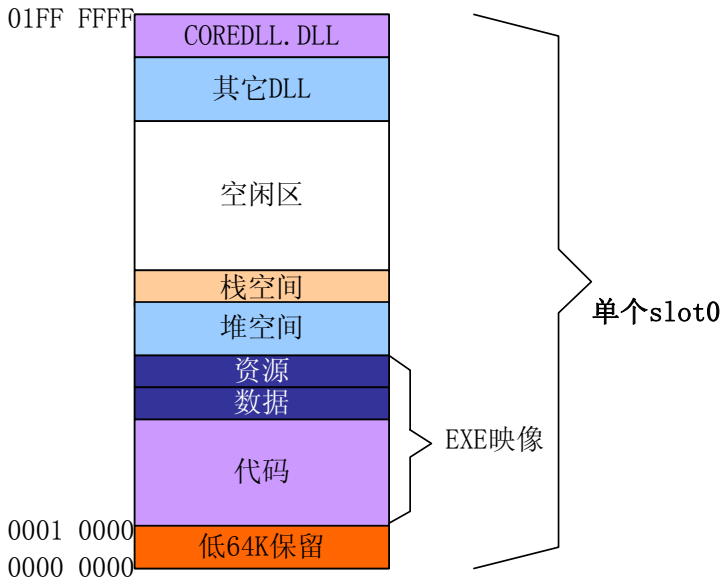


图 3-3内存槽slot0的结构

Fig. 3-3 The architecture of slot0

3.2 Windows CE 组件机制

3.2.1 Windows CE 组件特点

Windows CE在组件机制的实现上基本沿用了Windows的COM组件机制，COM组件除具备一般组件机制的特性外，对于Windows CE的COM组件还有它自身的特点，即COM组件的一些运行信息须要存放在系统的注册表中，组件在能够正确运行之前，必须到注册表中注册相关信息。此外，COM组件导出接口的描述是由类型库元数据提供，类型库被打包在组件的DLL文件中，但类型库信息要由系统的ole32.dll等COM库来解释。组件的创建需要通过组件类工厂来得进行。

3.2.2 Windows CE 组件元数据组织

Windows CE用idl（Interface Description Language）文件来描述组件接口及组件的其它一些属性要求，如组件的并发性限制等，idl文件中的信息在经midl编译后，会被编译为单独的tlb文件，打包在组件的可执行文件.exe或者.dll中，作为描述组件的元数据而成为组件文件资源的一部分。tlb中的信息可通过微软提供的一组接口ITypelib进行访问[45]。

3.2.3 Windows CE 组件实现原理

Windows CE注册表

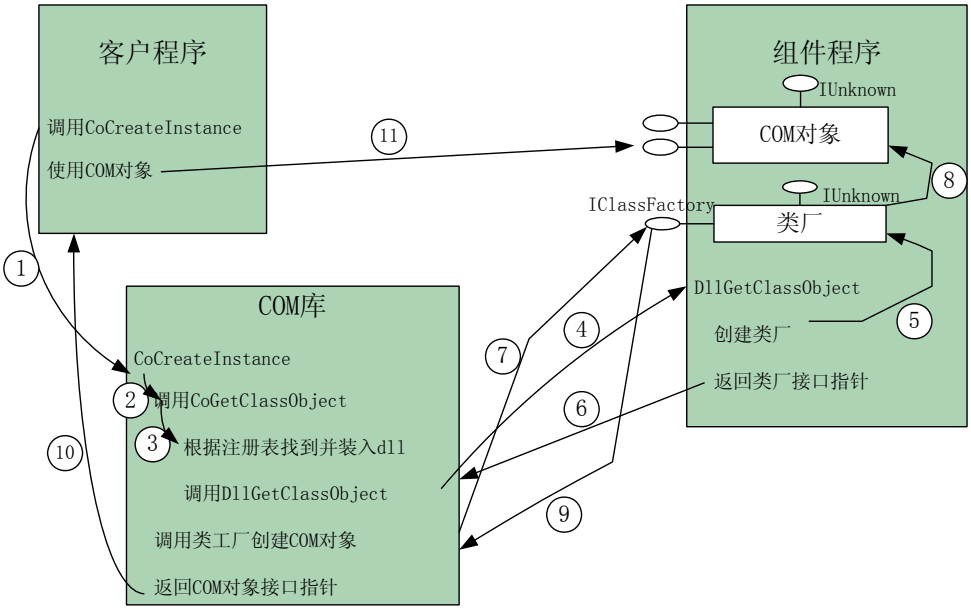
Windows CE的COM规范使用128位GUID来标识COM对象和接口，客户程序通过这些GUID值来创建COM对象并与对象进行交互。因为客户程序与组件程序是独立的，客户程序在创建对象进并不一定知道组件程序的确切位置，按照COM规范，客户程序通过COM库完成对象的创建工作。COM库则通过系统注册表所提供的信息进行组件的创建，而注册表信息中与COM创建有关的一个最重要的信息便是组件的确切位置信息，它告知客户在哪里可以找到相应的组件来加载运行。系统注册表是一个全操作系统范围公用的信息仓库，其中存储了所有的COM组件必要的信息。客户程序和组件程序都可以对系统注册表进行访问。

进程内组件的实现

组件分为进程内组件及进程外组件。进程外组件以exe程序的形式存在，它在被调用时有它自己的进程空间，而进程内组件则与客户程序处于同一进程空间。客户程序与进程外服务器进行通信时，并不直接进行，而是分析通过代理存根dll来实现。当客户进程要求创建一个进程外服务器时，COM通过注册表找到进程外服务器，在一个新的进程空间将之加载运行，并为它们启动代理存根dll，以满足客户程序与进程外服务器进程之间进行通信的需要。当然实际进程外服务器进程启动过程并不这么简单。我们这里主要以进程内服务器COM组件为例，来看看Windows CE系统上组件的实现过程。

Windows CE 的进程内组件dll都提供了DllUnregisterServer、

DllCanUnloadNow、DllGetClassObject及DllRegisterServer四个dll引出函数。DllCanUnloadNow用于卸载进程内组件dll，DllRegisterServer用于将进程内组件dll中的组件对象进行注册及，DllUnregisterServer则起到将组件对象的注册信息从注册表中删除的作用。而DllGetClassObject则对进程内组件对象的创建非常重要，这它负责向调用者提供本组件程序中的组件信息[16]。



① 客户调用CoCreateInstance以获取COM对象
② CoCreateInstance调用CoGetClassObject函数以获取类厂。
③ CoGetClassObject查找注册表并找到DLL程序加载到进程。
④ CoGetClassObject调用组件dll内提供的函数DLLGetClassObject。
⑤ DLLGetClassObject 函数创建类厂。
⑥ DLLGetClassObject 函数把类厂接口指针返回给CoGetClassObject函数。
⑦ CoCreateInstance从CoGetClassObject得到类厂接口后，调用类厂以创建COM对象。
⑧ 类厂创建COM对象。
⑨ 类厂将COM对象接口返回给CoCreateInstance
⑩ CoCreateInstance将得到的COM对象返回给客户
⑪ 客户直接调用COM对象

图 3-4　Windows CE 进程内组件的创建

Fig. 3-4 The creation of in-process componet on Windows CE

一般地，我们创建一个进程内组件对象是通过调用COM库CoCreateInstance，并将相应要创建的组件对象的clsid和iid作为参数传入就可。然而，

CoCreateInstance 其实是做了两步工作：一是通过调用 COM 库的 CoGetClassObject 获取组件 dll 的类厂对象；其次调用类厂对象的 CreateInstance方法生产出客户程序所想的组件对象来。但是，组件对象所在的组件dll在哪，它甚至可能还未加载到客户进程空间，CoGetClassObject又如获得组件dll的类厂对象呢？因此CoGetClassObject必须解决两个问题：一是加载组件dll进入客户进程空间，这通过CoLoadLibrary实现(CoLoadLibrary利用组件对象的clsid查找注册表，获知组件对象的位置信息，从而将它加载进内存)；一是得到组件的类厂对象，它便通过调用组件 dll 提供的引出函数 DllGetClassObject(DllGetClassObject存在于组件dll中，所以能够顺利获取该组件dll的类工厂)。图3-4 给出了进程内组件创建的大致过程。

3.3 Windows CE 系统平台分析

3.3.1 应用程序与 Windows CE 系统

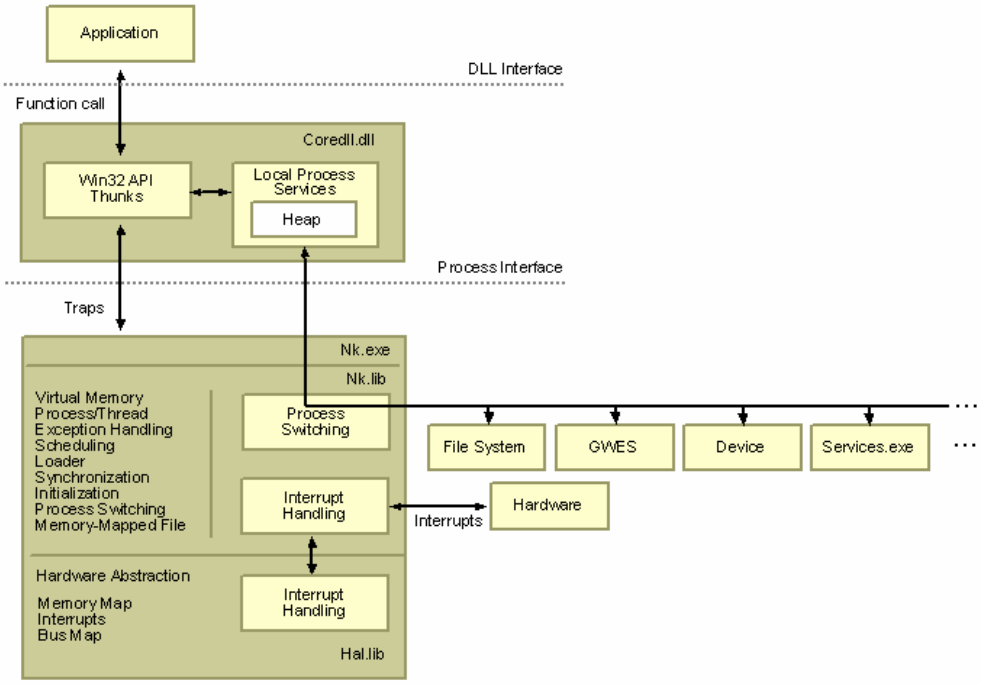


图3-5 Windows CE上应用与内核

Fig. 3-5 Application based on Windows CE and it’s kernel

与微软Windows系列的其它操作系统一样，Windows CE系统向应用程序提供的系统服务都通过系统API来实现，这些系统API服务都存在于动态链接库中。当一个应用程序执行时，相关系统服务的动态库便也被加载到进程空间。应用程序要求的内核服务需通过这些API传递到内核中。对于Windows CE系统而言，coredll.dll是一个最重要的一个系统平台级服务dll，它其中含盖了应用程序可能需要最基本服务接口，如C运行时函数，进程、线程的创建与管理接口等。对于组件程序，则相关的COM库dll来提供支持。总之，Windows CE上应用程序与内核之间的通信是通过一些系统平台dll来实现。如图3-5所示。

3.3.2　Windows CE 平台构成及其关系

平台主要构成

从编程角度来看，Windows CE有系统应用平台层支持超过1000个常用的Microsoft Win32 API和一些附加的编程接口，可用于开发应用程序。它们分布于系统平台的一系列动态链接库中。从用户应用能感知的功能角度看，Windows CE操作系统主要由下面几大功能模块构成[44]：

- **内核与组件服务模块：**这是 Windows CE 的核心模块，它向应用程序提供基本的系统服务。如进行进程、线程对象的创建与管理，中断、异常的处理，内存及各种资源的管理。提供临界区对象、事件对象、互斥体对象等同步机制，并对基本 C 运行时提供支持。提供在 Windows CE 上组件应用的支持，它对外表现为 coredll.dll、ole32.dll 及 oleaut32.dll 等动态链接库。
- **对象存储/文件系统服务模块:**Windows CE 的 RAM 的存储内存空间又被称为“对象存储”，包括文件系统、系统注册表、及数据库三种类型的数据。Windows CE 的文件系统包括三种类型：基于 RAM 的文件系统，基于 ROM 的文件系统和 FAT 文件系统。在 Windows CE 上，应用程序对文件系统的访问均通过 API 函数来进行。Windows CE 的数据库采用的是一种结构化存储方法，而其支持的注册表函数则是 Windows 上注册表函数的一个子集。
- **网络与通信模块：**Windows CE 提供了比 windows 桌面操作系统更丰富的通信支持，Windows CE 设备可以通过串行连接与其他 Windows CE 设备、PC、打印机等通信。支持 TCP/IP、FTP、HTTP 协议和 Winsock1.1 的一个子集，也支

持红外通信协议 IrDA、IrSock 和 IrComm 等。

GWES 模块：此模块即图形、窗口、事件子系统服务模块，是用户、应用程序、操作系统之间的图形用户界面。它对外主要表现为 Commctrl.dll 及 Commdlg.dll 等动态库。

主要模块依赖关系

通过对Windows CE各主要平台模块中函数引入引出关系的分析，得到Windows CE系统各主要平台模块之间的依赖关系如图 3-6所示。在众多的Windows CE系统服务模块中，动态库coredll.dll是一个非常基础的模块，其中提供了近两千个系统服务的API，涵盖了一个Windows CE应用系统基本需求。

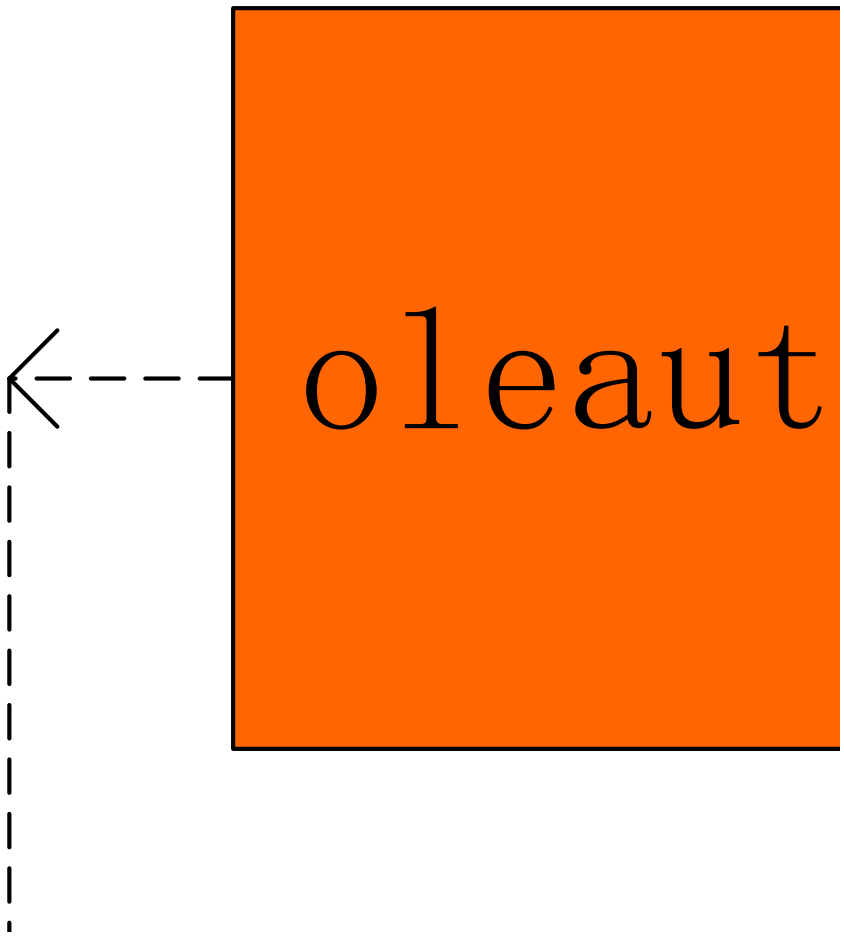


图 3-6　Windows CE系统平台模块依赖关系

Fig. 3-6 The relationship between Windows CE system platform modules

批注 [lyj14]: 建议：此图每类模块色调代表的含义应与“和欣”一致为好。

3.4 本章小结

本章对嵌入式操作系统Windows CE的体系结构及其特点作了简要介绍，并根据跨平台问题解决的需要，对其COM组件机制及其实现原理进行了比较详细的叙述，并最后从编程角度，在Windows CE操作系统的平台服务层面对Windows CE操作系统的平台主要有关模块的构成其它们之间的依赖关系进行了分析。

第4章　跨平台设计

4.1　操作系统的可执行模块

4.1.1 可执行模块入口点与文件格式

和欣操作系统和Windows CE操作系统都具有两种基本可执行单元：应用程序(表现为.exe文件)和动态链接库(表现为.dll文件)，它们被称为可装载的或可执行的模块。在用户概念上，应用程序入口点是main或WinMain(称之为主函数)，而对于动态链接库，也必须有一个入口点，DllMain是一个缺省的入口函数，它负责初始化和结束工作，每当一个新的进程或者该进程的新的线程访问DLL时，或者访问DLL的每一个进程或者线程不再使用DLL或者结束时，都会调用DllMain。但实际上对于一个操作系统而言，一个新的exe程序要运行，它就必须为之创建相应的进程，并对进程空间作一些初始化的操作，此后才进入用户的主函数，而当从用户的主函数退出后，又需要收回进程空间，这一系列过程，则是由用户所不知晓的mainCRTStartup函数或WinMainCRTStartup函数来完成。而对于动态链接库则需要知道是新进程还是新线程要访问它，或是有访问它的进程或线程退出，以及对于不同情况要怎样的处理，DllMainCRTStartup函数则在背后份演了这种角色。因此，对于跨平台问题的解决还需要考虑DllMainCRTStartup函数、mainCRTStartup函数或WinMainCRTStartup函数可能的函数调用问题的解决。

和欣操作系统上的可执行模块的文件结构均采用的是PE格式的文件结构，PE文件对可执行模块各种信息及各部分信息在文件中的位置进行了格式化的描述[51]。因此，不管是和欣的可执行模块还是Windows CE的可执行模块，Windows CE的装载器都能正常装载，在和欣操作系统上也是一样。

4.1.2 动态链接库工作原理

动态链接库（DLL，即Dynamic-Link Library）是操作系统的一种可执行模块，但它没有自己的堆栈，必须在调用动态链接库函数的程序环境下运行。动态链

接库不仅可以作为一个运行模块，它可以包括函数代码，而且可以包含程序以外的任何数据或资源（位图、图标等等）。动态链接库用于给应用程序提供函数或者资源[46]。

动态链接与静态链接是相对立的。静态链接的时候，可执行程序内包含了所访问的函数的代码，可执行程序占用的空间较大；但运行时，不需要其他模块支持。动态链接的可执行程序中不包含动态访问的函数代码，仅仅包含对它的参考，运行时需要其他模块（DLL）的支持。因此，对于常规的函数库，链接器从中拷贝它需要的所有库函数，并把确切的函数地址传送给调用这些函数的程序。而对于动态链接库，函数储存在一个独立的动态链接库文件中。编译器在编译应用程序时，链接过程并不把动态链接库文件链接到程序上。而是直到程序运行并调用一个动态链接库中的函数时，该程序才要求这个函数的地址。此时操作系统才在动态链接库中寻找被调用函数，并把它的地址传送给调用程序[49]。

动态链接库达到了复用代码的极限。动态链接库不用重复编译或链接，一旦装入内存，其中的函数可以被系统中的任何正在运行的应用程序软件所使用，而不必再将动态链接库函数的另一拷贝装入内存。制作动态链接库的目的之一是资源/代码二进制级共享[50]。因为，建立应用程序的可执行文件时，不必将动态链接库连接到应用程序中，而是在运行时动态装载相应动态链接库，装载时动态链接库被映射到调用进程的地址空间中。

动态链接库是这两个嵌入式系统极其重要的技术。它使得开发人员可以通过编写动态链接库，方便灵活的实现大型程序的开发，按自己的意愿对操作系统进行扩展。

4.2 跨平台相关问题

4.2.1 跨平台理念

本文对于跨平台的实施是源于如下两个理念：

1. 可执行模块在两嵌入式操作系统上二进制兼容：要求运行于一个嵌入式系统平台上的可执行模块不经重新编译，直接在另一嵌入试系统平台上二进制兼容运行。

2. 编译器无关性：可执行模块的开发者不论是谁，也不管其所用的编译器是什么，只要它基于同一操作系统的系统服务模块(即平台模块)来开发，那么这个可执行模块就能达到理念1的目标。

4.2.2 两平台的差异

通过前面对两嵌入式系统的一些分析，可以看到两系统之间存在如下方面的不同：

- 两系统平台虚拟内存映射机理不同：和欣系统的每个应用程序都有一个4GB的虚拟地址空间，其中有近2GB可供应用程序使用。而Windows CE则是整个系统共用同一个4GB的虚拟地址空间，而每一个应用程序本身的虚拟地址空间仅为32MB，因此应用程序的“大小”受到上限约束。理论上，和欣操作系统上可同时运行的进程数量是不受限制的，而在Windows CE上最多可同时有32个进程在运行。
- 两系统上组件实现不同：虽然在两个操作系统上组件的实现原理相似的，但Windows CE操作系统中，组件的实现过程需要的信息有赖于系统注册表的支持，而和欣系统中则没有类似注册表的东西，它的组件能够完全自描述，组件实现过程中所需的相关信息也全部来自于组件自身的元数据，并能够支持网上寻求构件动态下载。
- 进程线程同步机制等访问不同：和欣操作系统是一个完全组件化的操作系统，系统中所有的进程、线程及同步机制对象都是一个个鲜活的组件对象，组件是整个系统的灵魂，因此，对于这些对象的相关操作，实际上就是对相关对象接口方法的访问。而Windows CE系统上的这些对象非组件对象，对它们的访问则是通过传统的API方式来进行。
- 跨进程间的同步及数据共享问题：对于组件对象，只要能得到它的接口，就能访问组件对象的方法。和欣2.0系统实现了命名服务机制，在一个进程中注册命名服务对象，能在另一个进程中通过查找命名服务来得到这个对象的接口。正是通过命名服务机制，和欣操作系统能够实现进程间组件对象共享，因此对于进程间同步问题及数据共享问题，也都可以通过命名服务机制来实现,如提供命名同步机制对象，提供命名的共享内存对象等。而Windows CE

系统上进程间数据共享则可能通过映像文件的形式提供，进程间的同步也是通过相关同步机制API的访问来实现。

4.2.3 需解决的问题

要实现在一个系统平台上开发的应用，能够在另一个系统上二进制级兼容运行，就必须保证应用对系统的访问在另一个系统上的一致解决。也就是说，在一个平台上访问了系统平台某个DLL(动态链接库)的某个API，就必须在另一个系统上存在这样一个DLL以及对应的API,且API的名称、参数、调用方式等在两个平台必须是一致的；在一个系统提供了与应用相关的一些系统组件，则这些组件必须在另一个系统上得到一致解决，且这些组件接口提供的各方法在接口中的顺序，各接口方法的参数个数及类型，参数属性都应该在两个平台上是一致的。由于两系统存在的诸多不同，因此在跨平台问题，有如下问题需要妥善得到解决：

- 对系统对象/接口一致实现问题：如图4-1,图4-2所示，在一个操作系统上开发的应用程序，一经编译成可执行文件，它就与特定的系统服务模块产生依赖关系，那么在应用程序的加载运行过程中，对应的系统服务模块也必须能顺利地被加载运行。因此，要保障应用能跨平台顺利运行，首先必须保证这些系统服务模块在新平台的出现，并保障这些模块向应用程序提供的服务，在另一平台上，应用程序感觉不到任何差异。

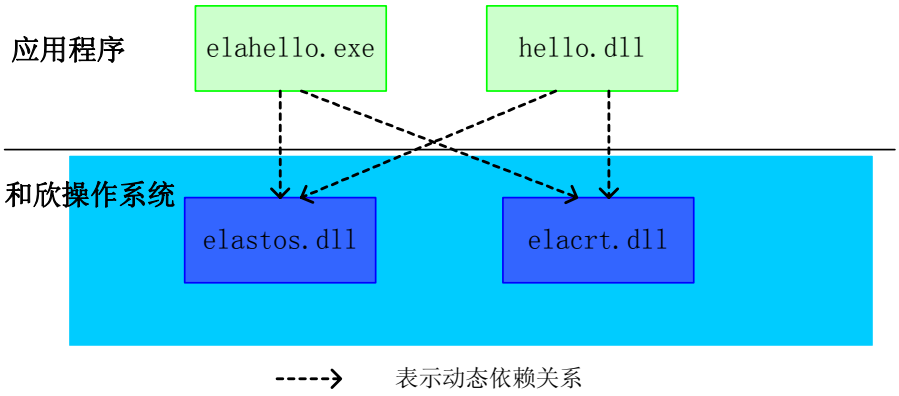


图4-1　和欣应用与系统服务模块关系

Fig. 4-2 The relationship between applications based on Elastos and it’s system service modules

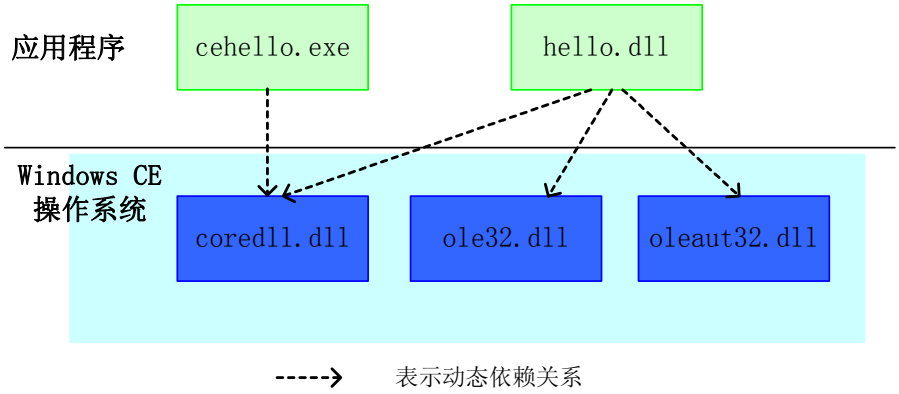


图4-2　Windows CE应用与系统服务模块关系

Fig. 4-2 The relationship between applications based on Windows CE and it’s system service modules

- 组件实现过程中相关信息的提供问题：和欣组件在Windows CE运行，最好的办法是仍借用Windows CE的组件实现机制，但这需要到Windows CE的系统注册表中注册和欣组件相关的信息,并提供Windows CE所需要的元数据。对于Windows CE上的组件要到和欣操作系统上运行，也有类似的问题需要解决。
- 虚拟内存问题：由于两系统上提供给应用程序的虚拟内存空间大小不一样，所以需要将这个问题纳入考虑范围，尤其是将和欣系统上的应用放到Windows CE上运行更需要好好考虑，因为它原来在和欣操作系统上所用空间是足够的，几乎可以说是不受限制的，但到了Windows CE却要受32MB虚拟空间所约束。
- 特殊硬件驱动的一致访问问题：嵌入式操作系统一般都用于特殊设备上，而这特殊设备可能有特定的一些硬件需要访问，我们知道操作系统对硬件的访问是通过驱动程序来进行的。而一个嵌入式系统上的应用又不可避免的需要操控硬件，因此，这就必须保证应用程序内部对硬件驱动的访问方式与原系统上一致。

批注 [lyj15]: ?

4.3 和欣2.0到Windows CE的跨平台设计

4.3.1 跨平台实现方案

实现方案

由4.2节的分析可知：可执行模块一经生成，它与操作系统某些具体的服务模块就存在依赖关系，如果不将可执行模块的源码，针对新的操作系统作适应性修改并重编译，那么就得在这些具体的系统服务模块上来寻求解决办法。此外，一个操作系统一旦定制完毕，它所提供的系统服务模块是相对稳定的，而这系统上的应用，不管中间经过多少环节，最终都需要这些服务模块的支持。有幸的是，和欣2.0系统和Windows CE的系统服务平台都是以动态链接库或组件库提供出来，动态库及组件库都可实现二进制兼容运行。因此对于和欣应用跨Windows CE平台运行，即然已经生成的应用要求不动，那么可以采取“瞒天过海”的办法，变变下面的系统服务模块，在Windows CE系统上提供一个和欣2.0系统服务模块的仿真层的方式来达到目标，这里称之为“和欣平台仿真层”，如图4-3所示。

批注 [lyj16]: ？

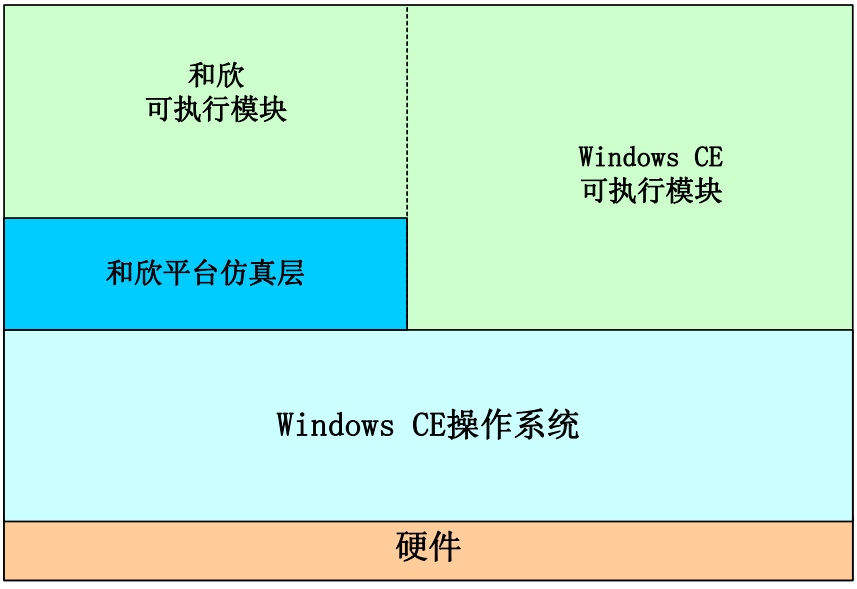


图4-3 和欣2.0到Windows CE的跨平台实现方案

Fig. 4-3 The scheme about cross-platform from Elastos 2.0 to Windows CE

要想在Windows CE上支持尽可能多的和欣应用，这个“和欣平台仿真层”涵盖范围就必须足够大，即需要仿真和欣2.0系统的所有系统服务模块。这些仿真

层的模块与和欣2.0系统上对应的模块一样，都是以动态链接库或组件dll的方式来提供。而且，对于仿真的每个服务模块，它们与和欣应用提供的服务接口与和欣2.0系统上所提供的完全一致，而仿真层模块的内部实现则完全基于Windows CE操作系统来施行。这样，当和欣应用在Windows CE上运行时，它的那些亲密伙伴——它所依赖的和欣系统平台模块在新的运行平台上都有，所以对于和欣应用而言，它的感觉上似乎仍然是在和欣系统上运行，丝毫不会觉察真正支撑它的是一个新的操作系统的平台环境。

因此，虽然在Windows CE系统上同时运行了和欣的应用及Windows CE的应用，便它们却仍然感觉是在各自原来所隶属的操作系统上运行一样。但这不等于和欣的可执行模块与Windows CE的可执行模块也一定会毫无关系。在4.1节谈到：可执行模块有两种，一种是exe可执行模块，一种是dll可执行模块。而dll可执行模块其中就有一种是组件dll。组件有一个特点，那就是只要它的接口被正确地获取到，那么，它就能为那个获取它的接口的执行模块服务。

例如，开发人员基于和欣系统开发了一个组件hello.dll，以及一个基于这个hello.dll的可执行程序elahello.exe，并基于Windows CE操作系统也开发了一个hello.dll，及一个基于这个组件dll的应用程序cehello.exe。基于两个系

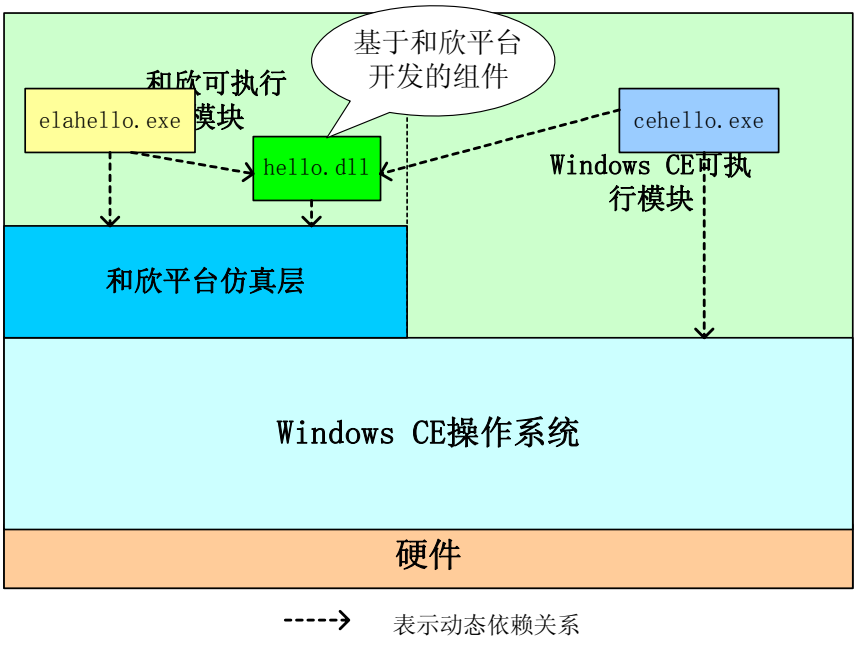


图 4-4　不同系统上的应用程序同用和欣组件

Fig. 4-4 Applicatons based on different OS call component based on Elastos

统上开发的hello.dll，它们所提供的接口都完全一致，并且组件的类ID、接口ID以及组件ID都完全一样。则不管是将基于哪个系统的hello.dll组件在Windows CE系统上注册后，两系统上的应用都能正确运行。但它们在运行中对组件，对系统平台的依赖关系则不一样。

当在Windows CE系统上将基于和欣平台的hello.dll注册后，两系统上的应用都能动态访问hello.dll中的方法，但hello.dll在运行中是与在Windows CE上的“和欣平台仿真层”存在直接依赖关系的，如图4-4所示。而当将基于Windows CE平台编译好的hello.dll注册后，两应用程序在访问hello.dll组件接口的方法时，不会发现有差异，但这个hello.dll所直接依赖的平台却是Windows CE了，如图4-5所示。

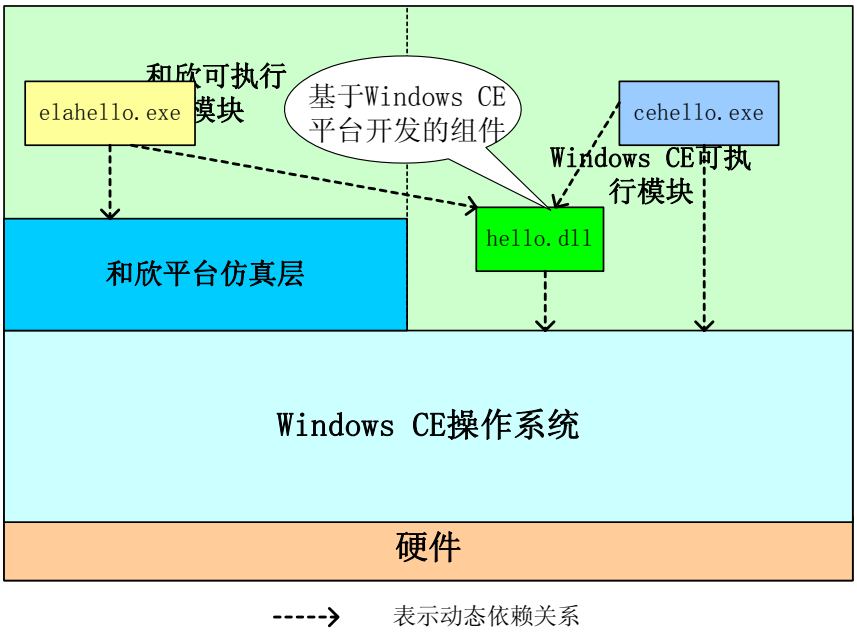


图4-5 不同系统上的应用程序同用Windows CE组件的示意图

Fig. 4-5 Applicatons based on different OS call component based on Windows CE

具体实施办法

基于某一操作系统的应用开发，最终都是居于这个系统向外提供的API或组件接口来进行。因此，和欣2.0系统在Windows CE上仿真层的实现，总体思想是向下对Windows CE的相关API进行封装，向上则以和欣上相应的模块的接口为参考，导出与之一致的接口。和欣2.0系统向应用提供三种编程接口：一种是C运行

时函数接口，一种是其它系统服务相关API,还有一种便是组件接口。

在和欣及Windows CE两个系统上都提供了标准的C运行时函数(它是大多数操作系统上很重要的部分)，在和欣2.0系统提供C运行时函数的是elacrt.dll这个动态库，而Windows CE则在动态库coredll.dll里提供了C运行时函数的实现。这些C运行时函数的函数名，参数类型及个数，函数调用方式，在两系统对相应动态库中都是完全一致的。所以仿真层中对于这种C运行时函数接口的实现，采用的是编译时函数一一映射机制，让生成的仿真层的elacrt.dll将对某一C运行时函数的访问动态重定向到对Windows CE上coredll.dll中对应C运行时函数的访问。这样，当在Windows CE上运行的和欣应用，要访问C运行时接口函数时，它最终实际访问的是Windows CE系统上coredll.dll中对应的C运行时函数。

操作系统向外提供什么API，是源于这个系统的设计理念的，和欣系统的设计理念与Windows CE的设计理念并不相同，和欣系统向外提供的自己的系统API，用于向和欣应用提供系统服务及系统组件的创建服务，这些API与Windows CE并不相同，因此对于这些API的实现，就需要针对一个API的功能，利用Windows CE提供的API来实现。

和欣系统与Windows CE设计理念一个非常重要的不同，便是它的组件化思想，如文件系统服务，一些网络服务等和欣系统的服务模块，是以组件方式来提供，甚至，内核的进程、线程、驱动等都是一个个组件对象。用户创建进程、线程、申请同步控制时，得到的也是一个个组件对象。因此，为了做到和欣可执行模块能在Windows CE上兼容运行，就要求它在Windows CE上的运行环境也提供这样的组件式系统服务。当和欣可执行模块在Windows CE上启动运行，或正在Windows CE上运行的和欣可执行模块创建进程、线程、或需要同步时，“和欣平台仿真层”便要创建相应的进程、线程、或需要的同步机制组件对象。仿真层中不管是进程组件对象，线程组件对象还是同步机制的组件对象，它们都要向下调用Windows CE的API接口，而向上以组件对象形式提供出来，其实质是对Windows CE的相关API以组件的形式包装了起来。在Windows CE操作系统上,真正的线程、进程及同步机制的创建等操作都是由Windows CE操作系统的相关服务模块完成的。和欣系统上进程、线程及系统服务有关的对象的创建均是由elastos.dll这个动态库引出相应的API来实现应用要求的。如要求创建互斥体对象时，通过调

用elastos.dll的函数STDAPI EzCreateMutex(/* [out] */ IMutex **ppIMutex)，便可通过指针ppIMutex得到互斥体对象接口。如果要求创建线程对象时，通过调用elastos.dll的函数STDAPI EzCreateThread(HRESULT (*pEntry)(void *), void *pArg,DWORD dwFlags,IThread **ppIThread)，并传入线程有关参数， 便可通过指针ppIThread得到线程对象接口。这些对象创建后，都是位于客户进程空间的。

和欣2.0系统还提供了一些跨进程服务，如建立跨进程的同步机制对象，建立进程间数据共享的共享内存对象等。这些服务要在Windows CE上得到兼容，就必须在仿真层也提供这样的服务，但这些对象是不能作为执行模块的进程内对象被创建的，否则就不能跨进程了。因此，对于这些特殊对象的创建要求，便引入

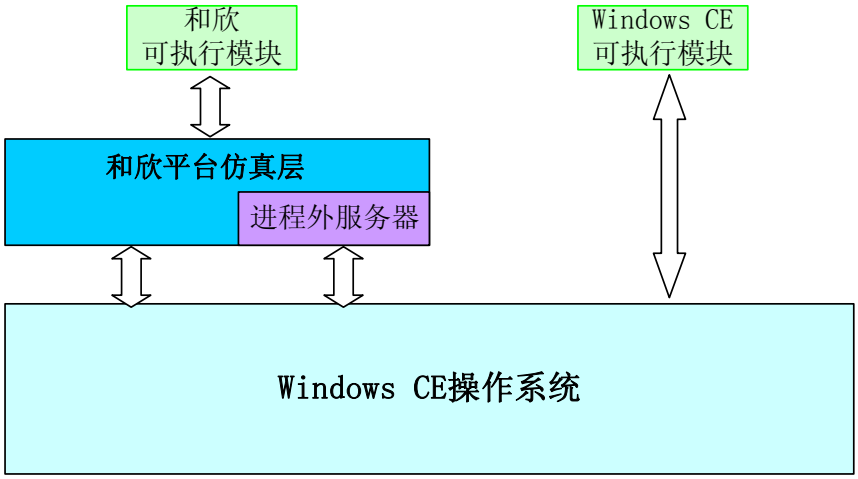


图4-6 进程外服务器参与跨平台服务

Fig. 4-6 cross-platform server with out-of-process component

了进程外服务器来解决。因为进程外服务器是一个独立的进程，当多个客户进程都要用到进程外服务器时，这个进程外服务器只有一个，且进程外服务器中创建的组件对象可以为多个进程所共享。而且进程外服务器中创建的组件对象，位于进程外服务器的进程空间内，不会占据客户进程空间。这种问题解决方式，既在一定程度上考虑了Windows CE上进程空间不能超限的要求，又实现了对象的跨进访问及被多个客户共享问题。进程外服务器为“和欣平台仿真层”的重要构成部分。有了进程外服务器的参与时，和欣应用与Windows CE之间的关系便如图4-6所示。进程外服务器虽然参与了应用的运行，但只与平台仿真层的elastos.dll模块产生联系，不直接与应用程序打交道。

4.3.2　在Windows CE上对CAR组件支持实现

由3.2.3可知，组件在Windows CE上能够正确地加载，并实例化出来运行，需具备两个先决条件：一是Windows CE能够找到组件，另一个则是组件dll必须导出类厂获取函数。Windows CE上运行和欣应用时，对和欣组件的加载运行最终需要Windows CE上的COM库来支持。在3.2.3,已经对Windows CE上的COM组件原理进行了很详细的分析，从中可知，系统注册表对COM库能否顺利加载组件起到非常重要的作用，因此，这需要开发了一个注册工具，用于将和欣组件加载所必须的信息注册到Windows CE的系统注册表中。

只要组件能够被Windows CE的COM库找到，并且组件dll能提供COM所需的获取类厂的导出函数DllGetClassObject，则一般组件就能够顺利得到使用。和欣组件中获取类厂的导出函数为_EzComDllGetClassObject，因此，为了能使组件在Windows CE上兼容运行，在组件生成时，对于_EzComDllGetClassObject函数在导出时名字也应该为DllGetClassObject，这一点在链接生成和欣组件时就已经得到实现。

4.3.3　CAR组件命名服务机制的实现

在2.2.5节对和欣的命名服务机制作了比较详细的介绍，的确它在和欣系统上进行跨进程通信，对象共享方面起了非常重要的作用。客户应用进行命名服务相关操作是通过和欣系统服务模块向系统上的客户应用提供的三个命名服务接口函数：EzFindService、EzRegisterService、EzUnregisterService来进行。在和欣2.0系统上，用户和内核都可以创建一个命名服务，但是命名服务的对象要向操作系统内核注册，由内核掌控的。而在Windows CE操作系统上的和欣系统服务模块仿真层，对于操作系统而言，相当于是它上面的一个应用，因此，在Windows CE系统上，和欣的可执行模块无法象在和欣系统上一样去向内核注册命名对象，因为Windows CE内核不提供这样的接口。

由于在和欣系统服务模块仿真层中实现的进程外服务器是唯一的，可以为任意多个和欣应用所访问，因此，可以将命名对象注册到进程外服务器中，让进程外服务在管理这些命名服务对象的功能上类似于和欣系统内核的功能。而要求系

统启动过程中就实现的一些可供用户进程可用的命名服务对象，则可以在进程外服务器启动时就让它们创建出来。

4.3.4 跨平台问题中驱动的解决

驱动是一个嵌入式应用不可避免要涉及的问题，在和欣操作系统上，每一类设备驱动程序都是一个命名服务的对象，且这些对象在系统启动后就在内核被创建好。客户应用要访问驱动时，可以通过系统服务模块的命名服务接口，获取相应驱动对象，利用接口方法进行访问，这也是客户获得设备驱动的唯一方式。数控有它专有的伺服设备，对它的访问要通过伺服驱动来进行。

为保证数控应用跨平台后的兼容运行，它在Windows CE系统上的伺服驱动也必须通过系统服务模块，让用户以命名服务的形式能访问到。而且必须保证当任务运行时，各任务访问的仍然是同一伺服驱动对象，也即伺服驱动为各任务进程所共享，因此，在Windows CE上伺服驱动对象的提供由仿真层的进程外服务器来实现，伺服驱动对象的唯一性也由进程外服务器来保证。和欣系统上，系统启动过程中就创建好了伺服驱动对象，这样客户应用就可以随时访问，而不必关心这个伺服驱动对象的创建问题，因此，在Windows CE上，仿真层的进程外服器启动时就创建好了伺服驱动对象，以供客户应用随时访问。

4.4 Windows CE 到和欣 2.0 的跨平台设计

4.4.1 平台实现方案

Windows CE的系统服务平台都是以动态链接库提供出来，动态库的特性决定了它是可以二进制兼容运行的。因此，与和欣跨Windows CE平台解决办法类似，对于Windows CE应用跨和欣平台运行，可以通过在和欣2.0系统上仿真Windows CE的那些服务模块的方法来实现，即在和欣2.0上提供一个“Windows CE平台仿真层”，如图4-7所示。

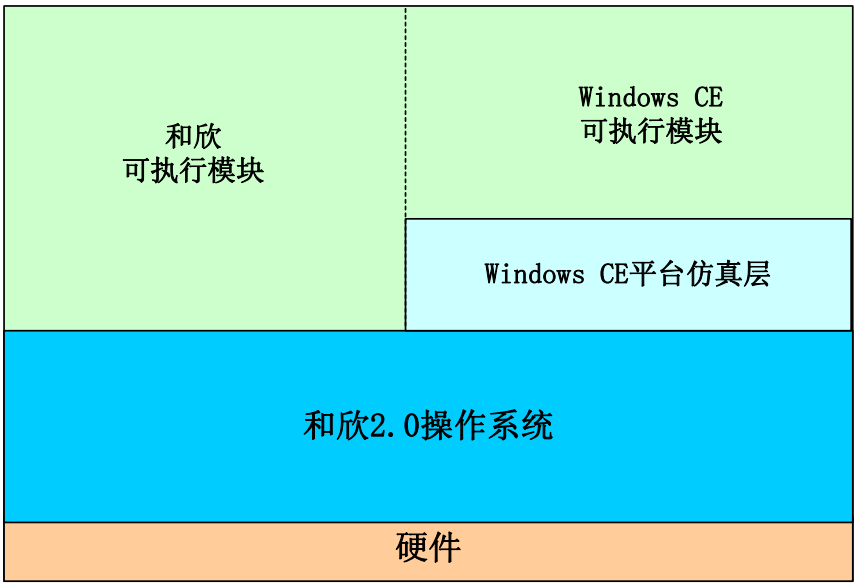


图4-7 Windows CE到和欣2.0的跨平台实现方案示意图

Fig. 4-7 The scheme about cross-platform from Windows CE to Elastos 2.0

要想在Windows CE上支持尽可能多的和欣应用，这个“Windows CE平台仿真层”涵盖范围就必须足够大。由3.3.2对Windows CE平台模块的分析及图4-2对简单和欣可执行模块与系统的依赖关系的分析可知，一般应用程序的运行有了coredll.dll的支持就可以，而当涉及组件需要运行时，则还需要ole32.dll oleaut32.dll的服务，当需要图形服务时，则需要commctrl.dll及commdlg.dll来支撑。因此，对于要在和欣2.0系统运行一些基本的Windows CE的应用问题，得在Windows CE平台仿真层中把这些服务dll的仿真出来。如果需要有网络功能，则还需要对网络相关的模块进行仿真。

4.4.2　在和欣 2.0 上对 COM 组件支持的实现

Windows CE上对COM组件的支持是通过COM库API来实现的，而在和欣上对组件的支持则是由和欣组件平台来实现，因此，在和欣上运行COM组件时，最终需由和欣组件平台来支持。但在和欣系统上创建CAR对象时需要的类ID的数据结构为EzCLSID，其定义如下：

```
typedef struct DECL_PACKED _tagGUID {
    UINT32 Data1;
    UINT16 Data2;
```

```
    UINT16 Data3;
    UINT8 Data4[8];
}   GUID;
typedef GUID CLSID;
typedef struct _tagEzCLSID {
    CLSID clsid;
    WCHAR *urn;
}   EzCLSID;
```

而Windows CE上创建COM对象时，类ID的数据结构则为CLSID，EzCLSID可以作为CLSID被访问，但CLSID却不能作为EzCLSID被访问。Windows CE在和欣2.0的仿真平台对于组件运行的这个问题的解决，可以通过建立一张表，表中为每一个COM对象的CLSID,建立一个对应的EzCLSID,在要求创建COM对象时,查找表中其CLSID对应的EzCLSID，再以得到的EzCLSID作为类ID，调用和欣组件平台的服务函数EzCreateInstance。

4.5 本章小结

本章分析了Windows CE及和欣2.0两操作系统上支持的可执行模块的一些特性，对在前两章的基本上对两平台上的差异性进行归纳总结，提出了本文所指跨平台的理念。并根据这一理念，以和欣应用跨Windows CE平台执行为主，给出了这一跨平台问题的解决方案，以及系统平台接口，组件机制的支持等一系列相关问题的解决对策。由于Windows CE应用跨和欣2.0系统运行总体解决思路是相似的，所以文中对于Windows CE平台到和欣2.0平台的跨平台设计只作了简要的说明。

第 5 章 跨平台性能分析

跨平台运行进,可执行模块在性能上会受有多大影响？这里对于和欣应用程序跨 Windows CE 平台运行分别在两台电脑上进行了一些测试，从和欣可执行模块在和欣 2.0 系统上运行,和欣可执行模块跨 Windows CE 平台运行,及在 Windows CE 上相似模块的运行在时间性能方面的差异进行比较分析。

两台测试电脑的 CPU 及内存配置情况分别为：PENTIUM III CPU 500MHz 128M 内存；PENTIUM III CPU 1000MHz 128M 内存。

性能测试则分两类情况来进行，一类是两系统平台都具有的接口函数的调用，一类则是组件方法调用。两系统上的相同系统接口函数的或相同组件方法的调用,能较好地反应出性能来。因为两系统平台都实现部分 C 运行时接口函数，且由于和欣可执行模块运行于 Windows CE 的和欣平台仿真层时，可执行模块对于这些 C 运行时函数的调用是直接转到对 Windows CE 上对应的函数调用，因些，对于这些函数进行跨平台运行时的时间开销能反应跨平台运行时对性能的影响。这里选取了三个 C 运行时函数来进行分析。而对于组件方法的调用分析则分别用和欣提供的编译工具编译用于和欣系统上运行的测试用组件及客户程序，用 Embedded Visual C 编译用于 Windows CE 上运行的测试用组件及客户程序。两种系统下用来测试的组件程序都仅提供简单的加(Add)、乘(Mul)、除(Div)方法及空方法(什么也不做，直接返回)，而客户程序则是对各方法调用的时间开销进行测试。

在时间开销的统计上分别是调用各操作系统上提供的系统时间获取函数来进行，由于 Windows CE 提供的系统时间获取函数得到的系统时间只精确到秒，所以相对精确度做不到很细，为了使时间开销能得到尽可能精确的反映，所以，在对函数或组件方法调用的时间开销上，分别从调用 100 百万次、400 百万次及 900 百万次的时间开销取平均值。

5.1 系统平台接口函数调用

表 5-1 及表 5-2 分别给出了对于三个 CRT 运行时函数 wcscmp、wcslen、abs 调用不同电脑上各种情况的测试结果。从表中可以看到：和欣测试程序运行于和欣 2.0 平台上统计到的三个函数的平均时间开销，与 Windows CE 测试程序运行于 Windows CE 平台上统计到的三个函数平均时间开销相差较大，这是由于两个系统上对于它们的实现不同而造成的；然而，对于和欣测试程序运行于 Windows CE 上仿真平台统计到的三个函数平均时间开销，与 Windows CE 测试程序运行于 Windows CE 平台，则在平均时间开销上相差并不多，即这种函数调用直接转换造成的时间多余开销并不多。

表 5-1 在 PENTIUM III CPU 500MHz 上测得的几个 C 运行时接口函数时间开销情况

Table 5-1 The time cost about C Run Time API calling on PENTIUM III CPU 500MHz

测试程序及其运行平台	函数	调用次数(百万次)	时间开销(毫秒)	平均时间开销(毫秒/百万次)
和欣测试程序运行于和欣 2.0 平台	wcscmp	100	15790	157.878571429
		400	63160	
		900	142080	
	wcslen	100	20980	209.842857143
		400	83950	
		900	188850	
	abs	100	3600	35.971428571
		400	14390	
		900	32370	
和欣测试程序运行于 Windows CE 上仿真平台	wcscmp	100	8000	76.428571429
		400	30000	
		900	69000	
	wcslen	100	12000	119.28571429
		400	48000	
		900	107000	
	abs	100	<1000	<1
		400	<1000	
		900	<1000	
Windows CE 测试程序运行于 Windows CE 平台	wcscmp	100	7000	72.142857143
		400	29000	
		900	65000	
	wcslen	100	11000	114.285714286
		400	46000	
		900	103000	
	abs	100	<1000	<1
		400	<1000	
		900	<1000	

表 5-2 在 PENTIUM III CPU 1000MHz 上测得的几个 C 运行时接口函数时间开销
Table 5-2 The time cost about C run time API calling on PENTIUM III CPU 1000MHz

测试程序及其运行平台	函数	调用次数(百万次)	时间开销(毫秒)	平均时间开销(毫秒/百万次)
和欣测试程序运行于和欣 2.0 平台	wcscmp	100	7910	79.071428571
		400	31630	
		900	71160	
	wcslen	100	10610	106.085714286
		400	42430	
		900	95480	
	abs	100	1800	18.014285714
		400	7210	
		900	16210	
和欣测试程序运行于 Windows CE 上仿真平台	wcscmp	100	4000	37.142857143
		400	15000	
		900	33000	
	wcslen	100	5000	57.142857143
		400	23000	
		900	52000	
	abs	100	<1000	<1
		400	<1000	
		900	<1000	
Windows CE 测试程序运行于 Windows CE 平台	wcscmp	100	4000	36.428571429
		400	15000	
		900	32000	
	wcslen	100	5000	57.142857143
		400	23000	
		900	52000	
	abs	100	<1000	<1
		400	<1000	
		900	<1000	

5.2 组件调用

表 5-3 及表 5-4 分别给出了对于进程内组件的四个简单方法调用，在不同电脑上各种情况的测试结果。从表中可以看到，与 C 运行时函数类似：和欣测试程序运行于和欣 2.0 平台上统计到的组件方法调用的平均时间开销，与 Windows CE 测试程序运行于 Windows CE 平台上统计到的组件方法调用平均时间开销相差较大，这仍是由于两个系统上对于它们的组件的实现机制上不同而造成的；然而，对于和欣测试程序运行于 Windows CE 上仿真平台的组件方法调用，与 Windows CE 测试程序运行于 Windows CE 平台组件方法调用，则统计到的平均时间开销上几乎没有什么差异，这是由于可执行模块跨 Windows CE 平台运行及在 Windows CE

上运行，其组件的支持，都是由 Windows CE 上的组件机制来提供支持的缘故，且组件方法与可执行模块位于同一进程空间里面。

表 5-3 在 PENTIUM III CPU 500MHz 上测得组件方法调用时间开销

Table 5-3 The time cost about component method calling on PENTIUM III CPU 500MHz

测试程序及其运行平台	组件方法	调用次数(百万次)	时间开销(毫秒)	平均时间开销(毫秒/百万次)
和欣测试程序运行于和欣 2.0 平台	Add	100	6800	67.95
		400	27180	
		900	61150	
	Mul	100	6840	68.442857143
		400	27380	
		9000	61600	
	Div	100	15590	155.871428571
		400	62350	
		900	140280	
	NoAct	100	5440	54.364285714
		400	21740	
		900	48930	
和欣测试程序运行于 Windows CE 上仿真平台	Add	100	4200	31.571428571
		400	12000	
		900	28000	
	Mul	100	3000	34.285714286
		400	14000	
		900	31000	
	Div	100	10000	95
		400	38000	
		900	85000	
	NoAct	100	1000	17.142871543
		400	7000	
		900	16000	
Windows CE 测试程序运行于 Windows CE 平台	Add	100	3000	30.714285714
		400	12000	
		900	28000	
	Mul	100	3000	34.285714286
		400	14000	
		900	31000	
	Div	100	10000	95
		400	38000	
		900	85000	
	NoAct	100	1000	17.142871543
		400	7000	
		900	16000	

表 5-4 在 PENTIUM III CPU 1000MHz 上测得组件方法调用时间开销

Table 5-4 The time cost about component method calling on PENTIUM III CPU 1000MHz

测试程序及其运行平台	组件方法	调用次数(百万次)	时间开销(毫秒)	平均时间开销(毫秒/百万次)
和欣测试程序运行于和欣 2.0 平台	Add	100	3400	34.028571429
		400	13610	
		900	30630	
	Mul	100	3430	34.285714286
		400	13720	
		9000	30850	
	Div	100	7810	78.064285714
		400	31230	
		900	70250	
	NoAct	100	2720	27.228571429
		400	10890	
		900	24510	
和欣测试程序运行于 Windows CE 上仿真平台	Add	100	2000	15.714285714
		400	6000	
		900	14000	
	Mul	100	2000	17.142857143
		400	7000	
		900	15000	
	Div	100	4000	47.142857143
		400	19000	
		900	43000	
	NoAct	100	1000	9.285714286
		400	4000	
		900	8000	
Windows CE 测试程序运行于 Windows CE 平台	Add	100	2000	15.714285714
		400	6000	
		900	14000	
	Mul	100	1000	17.142857143
		400	7000	
		900	16000	
	Div	100	5000	47.142857143
		400	19000	
		900	42000	
	NoAct	100	1000	9.285714286
		400	4000	
		900	8000	

5.3 本章小结

本章参考对 C 运行时一些函数及进程内组件方法调用，针对可执行模块跨平台运行与不跨平台运行在某些方面的时间开销差异地，对跨平台性能进行了分析，从中可以看出，对于组件方法调用，其性能几乎没什么影响，而对于系统接口函数的调用，则存在一此性能方面的影响，但也不大。通过各表的统计结果也可以看出，Windows CE 操作系统的整体性能，相对来说比和欣操作系统要优。

结论

对于当前日益受到重视的数控软件开放性的问题，本文力主在系统平台层面，谋求数控应用的最大开放性。虚拟机能够很好地解决跨平台问题，但应用程序的运行性能却也要受到很大的影响；软件重编译移植能解决应用程序在新的平台运行问题，却又未必能很好地实现数控软件开放性目标。如何使数控应用既做到跨平台又能实现更大的开放目标?本文借鉴中间件的思想，在操作系统平台层面来考虑这一问题的解决。

根据跨平台解决需求，本文先后对纳入跨平台问题的两个嵌入式操作系统(和欣 2.0 嵌入式操作系统，及 Windows CE 嵌入式操作系统)，进行了一些必要而细致的分析，并分析了可执行模块的特点及两系统下组件实现原理，分析出了各操作系统的主要构成模块及其基础模块，从而找出跨平台问题解决方案，并缕清具体实施办法。由于组件很大程度上满足了数控软件的开放性需求，因此，对于数控软件的跨平台，还特别需要解决的是组件软件跨平台问题。

本文以操作系统平台仿真的方式来达到使一个操作系统上的应用程序跨另一操作系统平台运行的目标。这种实现方式，可以实现最大范围的应用程序跨平台运行问题，即使得不管开发者是谁，用的是什么编译工具，其开发的应用都能通过这个仿真层在另一个操作系统上二进制兼容运行。这种跨平台的解决方式，使得其支持的跨平台应用可以不局限于数控领域，应该说其面向的是所有该系统上的应用。而对于组件跨平台的解决，则原则上由运行系统所实现的组件机制来提供支持，这使得对于组件方法的调用，不会由于仿真平台的存在而有过大的影响。

通过对跨平台性能的大量分析，可以看到对于两平台是都已经存在的相似接口函数，跨平台运行与不跨平台运行，在性能影响上不大，而对于进程内组件运行，由于组件与应用程序位于同一进程空间，进行的方法调用是在同一进程空间里进行，这与是否跨台运行没关系，因此对于组件的调用，性能上几乎没什么影响。

然而，这种跨平台的解决方式，对于轻量级的嵌入式操作系统尚可考虑，

因为，嵌入操作系统的提供的系统服务模块少而精，这相对于复杂的桌面操作系统而言，相对工作量要少。但这种解决办法是存在其不足的：一方面表现为，因为这种解决办法是基于某一操作系统的具体版本来的，当操作系统的版本升级时，新版本的操作系统的系统服务接口改动得越多，那么它在另一操作系统上的仿真要做的改动或者变动就越大，这是一种很被动的选择；另一方面，如果操作系统提供的系统服务模块很庞杂，则对其仿真则是一项耗时耗力的工程。

因此，如果在跨平台问题上要得到更大的自由，最好的解决方式则是开发出跨平台的中间件，用户应用基于中间件来开发，而不必考虑用的什么操作系统（这个只让中间件去考虑就行）。不过，它对于跨平台的支持则只局限于基于这个跨平台的中间件的应用。此外，对于数控等某一特定领域的应用，并不需要操作系统提供的所有功能，因此，这个中间件也可以放在数控领域需求的一个子集范围内考虑。不过，这种中间件的跨平台解决方式，需要解决一个如何使得应用仅基于中间件来开发的问题。

参考文献

1 《和欣2.0》资料大全 2004.8.30

2 Microsoft Windows CE.NET 4.2 Help version 4.0 2004.1

3 陈虎,韩至骏. 数控系统开放化的趋势与对策. 世界制造热核与装备市场. 2001 年第 4 期：8-11

4 童劲松,陆志强. 中德专家携手开发新一代数控系统. 机电一体化. 1999 年第 1 期:6-7

5 蔡葵. 中间件如何演绎“三层结构”好戏. 华南金融电脑. 2005 年第 1 期:75-76

6 王鹏、尤晋元. COBRA 与 DCOM 的比较[J]. 计算机工程. 1998 年 9 月:11-13

7 张凯龙. 传统OA的Linux中间件平台移植技术及其实现. 西北工业大学 计算机应用技术 硕士学位论文 中图分类号：TP393 2003年3月:14-31

8 车飞. 基于虚拟机平台的实时系统的设计与实现.浙江大学硕士学位论文 2004年2月:10-18

9 钟灿 钟本善 周熙襄. COM和COBRA的桥接与应用[J].电子科技大学学报. 2003年4月,第32卷第2期:188-191,206

10 陈克胜. 中间件：重新洗牌的机会. 中国电子商务. 2002年3月:44-46

11 尚海忠，朱培彦，王霞，徐家祥，陈涵生. 操作系统抽象层——一种支持跨平台的新技术 [J] .计算机工程. 2002 年 2 月,第 28 卷 第 2 期:109-111

12 Jianmin Zhu,Rong Chen,Guangnan Ni. Component-Based Middleware Platform For Grid Computing[J]. Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100080

13 柯海丰，吴明晖，应晶. 基于中间件的领域软件开发方法[J]. 计算机应用 2003 年第 8 期:54-56

14 闻达. 嵌入式开发要跨平台. 中国计算机报. 2003-10 第 1261 期

15 孔月萍. 软件移植方法与操作系统仿真[J]. 西安建筑科技大学学报(自然科学版). 2002年6月,第34卷 第2期:180-182

16 潘爱民. COM 原理与应用研. 2002.5 第 6 次印刷. 清华大学出版社：55-85

17 孙华志. Java 语言“与平台无关性”的实现[J]. 天津师范大学学报（自然

科学版）．　2002 年 12 月,第 22 卷 第 4 期:50-52

18 赵春云 郭 煦　金 戟　张久文. Java 语言及其虚拟机技术探讨[J]. 1998 年第 3 期:15-18

19 朱剑民　操作系统的跨平台技术

http://www.softcon.cn/expert/zhujianmin.asp　2003 年 12 月

20 Real Time and Windows Embedded　July 08, 2002

http://www.microsoft.com/windows/Embedded/community/experto/july2002/nframpton.asp

21 Martin Mellado[a,*], Eduardo Vendrell[a,1], Alfons Crespo[b,2], Pedro Lopez[c,3], Juan Garbajosa[c,4], Carmen Lomba[c,5], Klaus Schilling[d,6], Hubert Stutzle[d], Rudolf Mayerhofer[e,7]. Application of a real time expert system platform for flexible autonomous transport in industrial production. Computers in Industry. 38 1999 187–200

22 Hugues. wxPython a GUI Toolkit Talbot[J]. Linux Journal. June 2000, Volume: 2000　Issue: 74es

23 M.Di Santo, F.Frattolillo, W.Russo, E.Zimeo. A component-based approach to build a portable and flexible middleware for metacomputing[J]. Paralle Computing. 2002,28:1789-1810

24 Steffen Lipperts, Anthony Sang-Bum Park. An agent-based middleware-a solution for terminal and user mobility[J]. Computer Networks . 1999, 31:2053-2062

25 Brian E.Carpenter. Future applications and middleware, and their impact on the infrastructure[J]. Future Generation Computer Systems. 2003,19:191-197

26 蒲　维 邹益仁. 实时中间件动态高度算法的研究及应用[J]. 计算机工程与应用. 2003,28: 98-99,117 1002-8331-(2003)28-0098-02

27 郭长国，王怀民，邹鹏，苑洪亮. 实时中间件的研究与实现[J]. 电子学报. 2002年月12月,第12A期: 2094-2098

28 D C Schmidt, D L Levine, S Mungee. The design of the TAO real-time object

request broker[J]. Computer Communications. 1998, 21(4): 294-324.
29 Taran Rampersad wxWindows for cross-platform coding[J]. Linux Journal. July 2003,Volume: 2003 Issue: 111
30 曲红亭 申瑞民. COM / DCOM 与 COBRA 互操作的实现[J]. 上海交通大学计算机科学与工程系. 2000,30: 33-34，79
31 Kazuaki Ishizaki, Mikio Takeuchi, Kiyokuni Kawachiya, Toshio Suganuma, Osamu Gohda, Tatsushi Inagaki, Akira Koseki, Kazunori Ogata, Motohiro Kawahito, Toshiaki Yasue, Takeshi Ogasawara, Tamiya Onodera, Hideaki Komatsu, Toshio Nakatani. Effectiveness of Cross-Platform Optimizations for a Java Just-In-Time Compiler[J]. ACM SIGPLAN Notices October 2003,Volume: 38 Issue: 11
32 Thomas C. Brooke. Development of a distributed, cross-platform simulator[C]. Proceedings of the 2002 annual ACM SIGAda international conference on Ada. December 2002
33 Michael Babcock. The Importance of the GUI in Cross Platform Development[J]. Linux Journal. January 1998,Volume: 1998 Issue: 49es
34 Michael A. Cusumano, David B. Yoffie. What Netscape learned from cross-platform software development[J]. Communications of the ACM. October 1999 ,Volume: 42 Issue: 10
35 Masahiro Fujita[a,*], Hiroaki Kitano[b;1], Koji Kageyama [a;2]. A reconfigurable robot platform. Robotics and Autonomous Systems 29 (1999) 119－132
36 张晓刚. 中间件技术及其发展展望[J].微型机与应用. 2003.6:4-6，38
37 王晓东、彭兵、张际平. 基于中间件的开发研究[J]. 计算机应用研究. 2001.8:54-57
38 A.K.PFIFFER Inter CorPoration, 15201 NW Greenbrier Parkway CO1-01 Beaverton, OR, 97006, U.S.A Moving Paragon™ Operator System to a New Hardware Platform. Computers Math. Applic. 1998 Vol.32, No.7,pp.33-43
3. Niki Pissinou[a,*], Kia Makki[a], Birgitta Konig-Ries[b]. Mobile users in heterogeneous environments with middleware platform. Computer

Communications 26 (2003) 700－707

39 刘晓燕 张云生 J-J.Schwarz 李俊昌. 复杂实时系统软构件对象设计[J]. 计算机工程与应用. 2003.31:119-121,132

40 OverComing the Crisis in Real-Time Software Development. http://www.Object.com

41 李凌,李斌.开放式数控系统中组件重用技术研究.计算机与现代化. 2005 年第 5 期:1-3

42 赵新昱、郭兵兵 陈丈伟. COM 组件属性、方法的自动提取和调度[J].计算机工程. 第 26 卷 第 11 期,2000 年 11 月:12-13，95

43 吴丽贤 李丽 和力. 基于 COM 的自动化及其实现的几种方式[J]. 德州学院学报. 第 18 卷第 2 期,2002 年 6 月:74-77

44 田东风. Windows CE 应用程序设计. 机械工业出版社，2003 年 10 月:36-46

45 DON Box 著 潘爱民 译. COM 本质论.第四次印刷. 中国电力出版社， 2002 年 8 月：30-217

46 魏会贤 王培福 韩仲祥 陶晓燕. 动态链接库实现程序跨平台技术. 空军工程大学学报(自然科学版). 2000 年 6 月,第 1 卷第 2 期:46-48

47 王才 张一兵 周义仁 马福昌. 用 DLL 实现与语言无关的软件开发. 微计算机信息. 2001 年第 17 卷第 10 期:58-61

48 杨红颖. Windows环境下动态链接库的开发与调用. 烟台师范学院学报(自然科学版). 2000,16(1)：29-34

49 熊华 刘凤新 潘小莉. Windows动态链接库原理分析及其应用. 北京化工大学学报. 2004年,第31卷第1期:99-102

50 张世禄 彭磊. 利用动态链接库提高代码可重用性. 计算机应用. 2001年8月,第21卷第8期:239-240

51 肖俊武 杨海峰. Windows PE文件结构及其加密的研究. 湖北工学院学报. 2004年4月,第19卷第2期:24-26

52 和艳芬 翟海波 钟辉. PE文件格式解析. 交通与计算机. 2003年第5期,第21卷：109-112

53 Larry Hughes. Process migration and its influence on interprocess

communication[J] Computer Communications 21 (1998): 781－792

54 Baomin Xu[a,]*, Weimin Lian[b], Qiang Gao[c]. Migration of active objects in proactive Information and Software Technology 45 (2003) 611－618

55 Xavier Castellani[a,]*, Hong Jiang[b], Alain Billionnet[a]. Method for the analysis and design of class characteristic migrations during object system evolution. Information Systems. 26 (2001) 237－257

56 Kontogiannis, K.[1], Martin, J.[2], Wong, K.[2], Gregory, R.[3], M uller, H.[2] and Mylopoulos, J.[3]. Code Migration Through Transformations: An Experience Report

57 Kazuyuki Shudo*, Yoichi Muraoka. Asynchronous migration of execution context in Java Virtual Machines. Future Generation Computer Systems. 18 (2001) 225－233

58 杨 俊 皮丽娜. 传统 Win32 应用程序向 Windows CE 平台移植的研究[J]. 微型计算机与应用. 2002 年第 10 期:16-18

59 游有鹏. 开放式数控系统关键技术研究. 南京航空航天大学 机械制造及其自动化 博士学位论文 中图分类号: TP273

60 张凯龙 谷建华 盖玲兴 周兴社. Win32应用到Linux的跨平台移植技术研究. 微电子学与计算机. 2004 年第 11 期：102-106

61 Ruben Pinilla, Marisa Gil. JVM:platform independent vs. performance dependent*. Computer Architecture Department, Technical University of Catalonia c/Jordi Girona, 1-3, Campus Nord, 08034 Barcelona, Spain

致谢

本论文是在导师刘燕军老师的悉心指导下完成的。刘老师渊博的知识、严谨的治学作风和对问题的解决方法将使我终身受益。在我完成课题期间，刘老师在我的科研和生活上都给予了大力支持，通过几年来刘老师对我的悉心指导，使我学习到科学的研究方法和问题解决能力,从而使我的课题得以顺利完成。

此外，我课题研究的顺利完成，还得益于科泰公司 CEO 陈榕老师的热心指导，公司副总周宏乔的帮助与支持，苏翼鹏，梁宇洲，王晨辉，刘亚东，陈榕辉，以及和欣 2.0 组研发团队其他人员的指点与帮助，在此一并向他们致以衷心感谢！

同时，对在我课题完成过程中提供了很大帮助的哈达等同实验室成员表示诚挚的谢意。

在我的的求学历程中，我的父母付出了无数的汗水与心血，他们的鼓励、关怀与奉献，给了我一次又一次前进的力量与勇气。父母为我所做的一切，无法用言语来表达！