

和欣嵌入式系统命名服务机制研究

王志强, 陈 榕

(同济大学 基础软件工程中心, 上海 200092)

摘 要:“和欣”操作系统是基于 CAR 构件技术、支持构件化应用的嵌入式操作系统。“和欣”命名服务机制作为 CAR 构件的运行环境重要组成部分,在“和欣”中有十分重要的地位。命名服务屏蔽资源的位置特征,由此可以实现资源的透明访问、构件的动态升级和更新。文中分析了“和欣”命名服务的原理、特点及实现,并进一步深化了对面向服务的软件体系架构理解,把面向服务的体系架构引入到了嵌入式领域内,可以提高嵌入式领域内软件开发的灵活性和开发效率。

关键词:命名服务;“和欣”操作系统;CAR 构件

中图分类号:TP311.5

文献标识码:A

文章编号:1673-629X(2007)12-0001-04

Study of Named - Service Mechanism on Embedded OS of Hexin

WANG Zhi-qiang, CHEN Rong

(System Software Center, Tongji University, Shanghai 200092, China)

Abstract:“Hexin” is an embedded operating system based on CAR component technology and it supports the application based on component. As an important component of run-time environment of CAR component, “Hexin” named-service occupies an important position on the “Hexin” system. The named-service screens the positional characteristic of resource, as a result, can implement the transparent access of resource and dynamic upgrade and update of the component. Discusses the principal, characteristic and implementation of “Hexin” named-service. It deepens the understanding of services-oriented software architecture and brings such a software architecture into the embedded system, which can improve the agility and the efficiency of software development.

Key words: named-service; “Hexin” operating system; CAR component

0 引 言

“和欣”操作系统是 863 计划的“基于中间件技术的因特网嵌入式操作系统及跨操作系统中间件运行平台”的重要成果,是一个基于构件的微内核现代操作系统^[1]。“和欣”操作系统与其他宏内核或微内核操作系统的最大区别就是微内核模型与基于构件技术的充分结合,形成了“和欣”操作系统的灵活内核架构模型。基于构件的“和欣”操作系统的服务是以 CAR 构件的形式来提供的(CAR 是 Component Assembly Runtime 的缩写,它规定了一组构件间相互调用的标准,使得二进制构件能够自描述,能够在运行时动态链接)^[2]。“和欣”命名服务机制属于 CAR 构件技术的一部分, CAR 构件技术通过命名服务机制提供一种在系统内部发布、获取、使用 CAR 构件的方法。“和欣”操作系

统采用了命名服务机制,把构件对象和一个逻辑名字相关联,屏蔽了提供服务的构件对象的位置信息,用户只需通过逻辑名字来调用构件对象所提供的服务,实现了用户对构件服务的透明访问^[3]。同时它也为“和欣”系统提供了更大的灵活:如动态更新构件。和欣操作系统是面向 SOA 设计的,这种设计的基石也正是命名服务机制。从某种程度上说和欣的“WEB 服务”也是命名服务机制在网络上延伸^[4]。

1 命名服务机制

1.1 “和欣”命名服务及其特点

顾名思义,命名服务就是“根据指定对象的名字来进行服务”,其实质是一种以字符串为标识的服务^[3]。它为客户端程序提供了一种查询机制,可以通过字符串形式的名称查询在服务器端有没有某种服务,如果有就可以使用,如果没有服务器端会返回错误通知,告知客户端没有相应的服务。服务的使用过程屏蔽了列集/散集等具体调用细节,向用户提供的是完全透明的服务^[2]。命名服务机制属于 CAR 构件技术的用户接

收稿日期:2007-02-27

基金项目:国家“863”计划重大资助项目(2001AA113400)

作者简介:王志强(1981-),男,河南南阳人,硕士研究生,研究方向为系统软件支撑技术;陈 榕,教授,博士生导师,研究方向为网络嵌入式操作系统、构件技术。

口部分,也是 CAR 构件运行环境的组成部分。

远程服务构件在创建的时候会向内核注册相关信息,并建立存根,称为 Stub,远程用户获得构件指针的时候在自己进程空间建立代理,称为 proxy^[5]。远程服务构件在内核注册的信息代表了该构件对象的存在,通过一个内核对象 Object 代表某个构件的注册信息,通过此信息,可以找到相关建立代理所必须的信息,另一方面,也可通过这些信息找到该远程构件以及构件服务相关信息。

命名服务包含两部分:命名服务 Server 端和命名服务 Client 端。Server 端获得某个构件的接口指针后,调用系统 API 函数 EzRegisterService 向系统内核注册自己的服务接口,它也可以选择合适的时间注销命名服务并释放资源。Client 端调用系统 API 函数 EzFindService 函数获得相应的构件服务。图 1 显示了命名服务的工作流程。

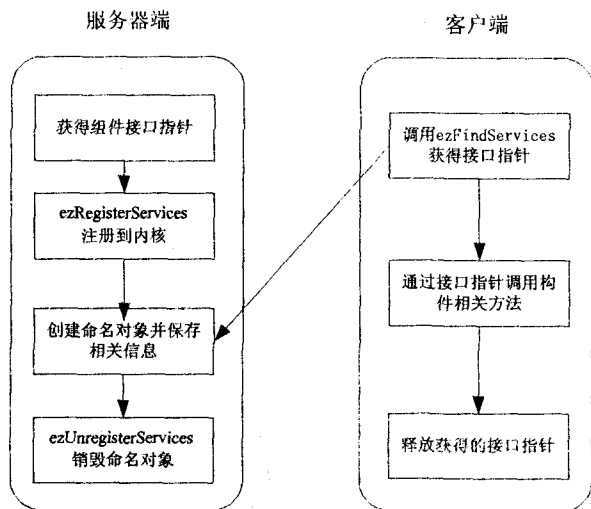


图 1 命名服务工作流程

命名服务具有如下特征:

(1) 用户和内核都可以创建一个命名服务,命名服务 Client 端可以透明地获得服务构件接口指针,而无需考虑其所在位置空间的不同。

(2) 命名服务 Server 端获得的构件指针可以是在 Server 进程空间内创建并运行的构件,可以是系统内核以接口形式提供的服务,也可以是在其它进程所实现的构件服务。

(3) 客户程序通过命名服务获得服务构件的接口指针后,其与服务构件交互的过程完全独立于命名服务。服务构件退出,则命名服务不再有效。

(4) 命名服务也是一个普通的构件,提供服务的构件可以在一个合适的时间点停止服务。所有适应于 CAR 构件生命周期管理的操作和语义亦适应于命名服务。

(5) 相关服务构件动态更新或者升级,不会影响通过命名服务获取该服务的客户程序,它无需改动,无需重新编译。

1.2 命名服务机制的具体分析

在“和欣”系统中,命名服务处于内核之中,有利于提高系统效率。命名服务机制的具体实现包含三个部分:服务的建立、服务的获取,以及服务的注销^[2]。

1.2.1 命名服务的建立

图 2 给出了命名服务建立的流程图。远程 CAR 构件在创建的时候会向内核注册相关信息,并建立存根,称之为 Stub,远程用户获得构件指针的时候在自己进程空间建立代理,称之为 proxy。远程 CAR 构件在内核注册的信息向其它用户表达了该构件对象的存在,通过一个内核对象 Object 代表某个构件的注册信息,通过对象 Object,可以找到建立代理所必须的信息,找到该远程构件以及构件服务相关信息。

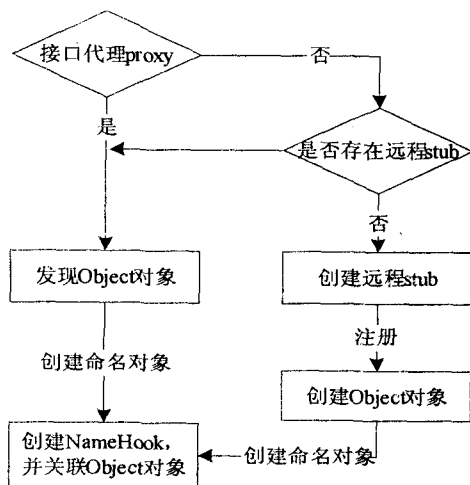


图 2 命名服务建立流程

在客户端,用户通过 API 函数 EzRegisterService (EzStr esName, IUnknown * pService) 来建立一个命名服务。参数 pServer 指要注册的服务构件的接口指针, esName 则代表要与该服务相关联的逻辑字符串。当注册成功后,客户程序可以通过该字符串获取相应的服务。参数 pServer 是远程服务接口指针或者本地接口指针。当它是远程构件接口指针时,其实质是接口代理指针,过接口代理找到相应的 Object,创建命名对象 NameHook,将二者关联起来;当它是本地对象接口指针时,则创建相应的存根 Stub,向内核注册相关信息,生成 Object 对象,创建命名对象 NameHook,将二者关联起来。

1.2.2 命名服务的获取

在内核中,创建命名对象 NameHook 并绑定名称之后,系统中的 NameHook, Object, Stub 之间存在一一

对应关系,因此可以通过字符串找到相应的 NameHook,通过 NameHook 找相应构件对象的 Object,通过 Object 获得足够建立 Proxy 的信息。图 3 表示了通过命名服务机制来获得远程构件服务的具体流程。

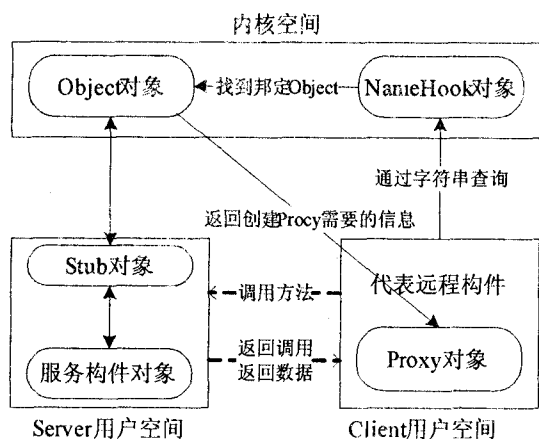


图 3 远程构件服务流程

(1) 用户调用 API 函数 EzFindService (EzStr esName, IUnknown ** ppService), 该函数传入两个参数: esName 代表名字服务, ppServer 则是要返回的构件服务指针。通过系统调用 HRESULT SysResolveCommon (const wchar_t * pszName, oid_t * pRet, CLSID * pClsid, UINT * puIndex, CiclassEntry ** ppClsInfo, DWORD * pdwContext) 陷入内核, pszName 为传入的参数, 其它为返回参数。

(2) 用 pszName 在全局的 NameHook 对象 Hash 表中查找相关的 NameHook 对象, 并由此而找到相应的 Object 对象。

(3) 根据 Object 对象的数据项进程对象指针确定服务的类型, 通过服务端的存根 (Stub) 对象来获取服务构件指针。

(4) 返回创建 Proxy 的相关信息, 在客户端建立相应的类对象代理 (Proxy), 将服务构件指针返回给客户程序, 至此整个获取服务过程完毕。

1.2.3 命名服务的注销

命名服务的注销是调用 EzUnregisterService 实现的。它的实质是取消 NameHook 对象与相应 Object 对象的关联, 并销毁 NameHook。取消关联的同时, 释放相关构件指针, 禁止客户通过命名服务获得构件服务。

其流程如下:

(1) 命名服务提供者调用 EzUnregisterService (EzStr esName) 函数, 通过系统调用 SysUnregisterCommon (wchar_t *) 陷入内核。

(2) 查找字符串 esName 相对应的 NameHook。

(3) 销毁该 NameHook 对象, 并释放存根指针。

(4) 取消 Object 与 NameHook 关联, 返回。

由于命名服务和相关联的服务构件可能是不同用户构建, 服务构件可能会提前退出, 并销毁相应的 NameHook 对象和 Object 对象。用户此前获得的构件指针失效, 若调用相应服务, 则内核返回一个错误值, 以告知用户。

1.3 “和欣”命名服务和微软 COM 的名字对象机制的比较

COM 的名字对象机制, 是一种可以把对象名字解析到这些名字所关联的对象上的机制。它支持三种激活方式: 绑定到类对象、绑定到新的类实例, 以及绑定到保存在文件中的永久对象^[6]。微软的名字对象对外提供了统一的接口函数 CoGetObject()。分析 CoGetObject 函数的实现可以看出: 它将创建对象和建立对象和名字的绑定的两个过程合在一起, 当不存在字符串所指定的对象或者实例的时候, 它将创建一个对应的新的对象或者实例来满足用户的要求。因此用户不能指定特定的类实例进行绑定, 对于绑定的类对象来说, COM 规定组件的类对象必须实现 IParseDisplayName (对象生成接口) 接口才能使用名字对象绑定类对象的功能; 对于绑定到保存在文件中的永久对象, 则组件本身必须实现 IPersistFile (文件名 -> 对象传递接口) 接口, 名字对象机制的使用对组件本身的实现有限制和需求。它也不支持将一个远程接口代理和字符串绑定的功能。

“和欣”的命名服务机制, 支持远程接口代理, 可以实现字符串与对象的绑定, 允许将任意接口代理与字符串绑定, 因此组件服务和命名服务可以由不同用户提供, 更有利于组件服务的分发和使用^[1]。“和欣”命名服务机制也提供了一种服务与查询分离的方法, 提供查询服务的系统可以本身不提供服务, 而是由其他本地程序或远程程序提供。

1.4 “和欣”命名服务机制的整体特征

“和欣”命名服务从整体上看有三个明显的特点, 同时也是它的优势所在: 比较简单、安全性好、易于扩展。

(1) 简单易用。命名服务是“和欣”系统的基础性设施, 也是 CAR 构件的精髓技术之一, 功能十分强大。但是, 它的实现才仅仅有四个简单并且易于被用户理解的 API 来完成。并且允许将系统提供的各种服务接口, 比如进程, 线程, module, 同步对象等与对应的字符串绑定, 其它进程可以通过命名服务机制非常方便地获得该进程, 从而能很方便地实现进程间通讯, 扩展了系统的功能。

(2) 安全性好。从安全性来看, 用户可以通过将一个信任度不高的服务启动在一个单独的进程中, 并通

过命名服务机制获得,这样就通过进程地址空间机制,隔离了服务与用户,同时服务之间的数据交换可以经过系统的构件平台数据交互机制的检测。

(3)易于扩展。从扩展性来看,用户可以通过在保持接口定义不变的情况下,修改服务程序代码,升级服务程序;另一方面用户亦可以通过提供新的接口,来扩展新的功能,新的用户可以利用新的接口,而旧的用户则不会产生影响,其代码可以不经修改、不经重新编译,正常运行。用户通过 CAR 构件方式实现的程序,都可以通过命名服务机制的方式,提供给其它远程用户。

2 命名服务编程示例

在下面的例子中使用 dll 形式的 CAR 组件,目的是通过命名服务在控制台下打印一个字符串(限于篇幅,仅列出大体框架,进行了简化处理)。用客户端程序来调用这个组件。在 hello.car 文件中组件,实现类、接口以及方法的声明表示如下:

```
module Hello //组件 Hello
{
    interface IHello { // 接口 IHello
        Hello([in] EzStr inStr); //方法 Hello
    }
    class CHello //实现类 CHello
    {
        interface IHello;
    }
}
```

在 CHello.h 以及 CHello.cpp 文件中声明了 C++ 类型的 CHello 对象,并实现接口方法 Hello。CHello.h 文件中主要代码表示如下:

```
class CHello: public _CHello //CHello 定义
{
public:
    CARAPI Hello(
        /* [in] */ EzStr inStr);
};
```

CHello.cpp 文件中主要代码表示如下:

```
DECLARE_OBJECTFACTORY(CHello)
ECODE CHello::Hello //Hello 方法的实现代码
/* [in] */ EzStr inStr)
{
    printf("%S\n", (wchar_t *)inStr);
    return S_OK;
}
```

在 server.cpp 中,将创建一个 Hello 的组件,并将其注册为以“hello”为标志的命名服务,然后通过系统 API 函数 EzCreateEvent 获取一个内核 Event 对象服

务,并将其注册为“event”为标志的命名服务,服务使用完毕,注销服务,并释放相关资源。程序主要代码如下:

```
CHelloRef cHello; //声明接口智能指针
IEvent * pEvent; // 声明 Event 接口指针
//在本进程空间内创建一个 Hello 组件
HRESULT hr = cHello.Instantiate();
//注册命名服务,将 CHello 构件与字符串“hello”绑定
hr = EzRegisterService(EZCSTR("hello"), \
    (IHello *) (cHello));
//通过 API 函数获得一个内核 Event 对象服务
hr = EzCreateEvent(true, false, &pEvent);
//注册命名服务,将 Event 构件与字符串“event”绑定
hr = EzRegisterService(EZCSTR("event"), pEvent);
//让该进程等待
pEvent -> Wait(NULL);
//被其它进程唤醒,等待结束,注销服务,释放资源
EzUnregisterService(EZCSTR("event")); //注销 Event 服务
pEvent -> Release(); //释放 pEvent 指针
//cHello 为对象智能指针,所以程序退出,它会自动释放对象资源
```

```
EzUnregisterService(EZCSTR("hello")); //注销 hello 服务
```

在 client.cpp 中,将通过 EzFindService 函数使用以“hello”以及“event”为标志的命名服务,通过命名服务,找到与名字绑定的相关组件服务,并进行调用,调用完释放掉相关的组件服务指针。主要代码如下:

```
IHello * pIHello; //声明一个 IHello 接口指针
IEvent * pIEvent; //声明一个 IEvent 接口指针
HRESULT hr;
//通过字符串“hello”找到相关组件服务
hr = EzFindService(EZCSTR("hello"), &pIHello);
//远程调用 Hello 方法,打印“hello, world”
pIHello -> Hello(EZCSTR("hello, world"));
pIHello -> Release(); //释放获取的 IHello 接口指针
//通过字符串“event”找到相关组件服务
hr = EzFindService(EZCSTR("event"), &pIEvent);
//通过 notify 方法唤醒 server 进程
pIEvent -> Notify(0);
pIEvent -> Release(); //释放获取的 IEvent 接口指针
```

3 总结

“和欣”嵌入式操作系统是基于 CAR 构件技术、支持构件化应用的操作系统,是国家 863 计划支持的 TD-SCDMA 的操作系统标准^[2]。利用命名服务机制,“和欣”操作系统可以传递共享内存指针以及一些同步对象指针,从而实现了跨进程的进程间通讯,用户可以很方便地获取到“和欣”操作系统动态装载的网络系

(下转第 7 页)

$$\textcircled{2} \quad f(x) = \frac{H_{i1}}{H_{i2}}$$

其中 H_{i1} 和 H_{i2} 为边缘方向直方图 H 中特定角度对应的直方图纵坐标取值,比如水平、垂直或对角线方向。

对于彩色图像,可以对其 RGB 通道分别求边缘方向直方图作为候选一维特征。



图 2 人脸灰度样本

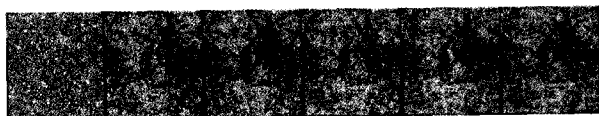


图 3 梯度方向区分图



图 4 基于方向直方图的 Adaboost
在测试集上的测试效果图

区分图(左起依次是将梯度划分为 4,8,16,32,64,128 个子集时梯度方向区分图,可见子集数越多,区分度越好)。图 4 是对测试集的检测结果。实验证明,本算法优于传统基于色彩的训练算法,具有更高的准确率,而速度与之相当。

参考文献:

- [1] Zhao W, Chellappa R, Phillips P J. Subspace Linear Discriminant Analysis for Face Recognition[R]. CAR - TR - 914. USA:Center for Automation Research, University of Maryland,1999.
- [2] 边肇祺,张学工. 模式识别[M]. 第 2 版. 北京:清华大学出版社,2000.
- [3] Wiskott L, Fellous J M, Kruger N, et al. Face recognition by elastic bunch graph matching[R]. IR - INI 96 - 08. Germany: Institut fur Neuroinformatik, Ruhr - Universitat Bochum,1996.
- [4] Fang Yuchun, Tan Tieniu, Wang Yunhong. Fusion of Global and Local Features for Face Verification[C]// IEEE International Conference on Pattern Recognition (ICPR). [s.l.]:[s.n.],2002.
- [5] Phillips P. Support Vector Machines applied to face recognition[R]. NISTIR 6241.[s.l.]:[s.n.],1998.
- [6] Gun GuoDong, Zhang HongJiang. Boosting for Fast Face Recognition[C]// IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real - Time Systems(RATFG - RTS'01), Conjunction with ICCV2001. [s.l.]:[s.n.],2001:96 - 100.
- [7] Freund Y, Schapire R E. A Decision - theoretic Generalization of On - line Learning and an Application to Boosting[J]. Journal of Computer and System Sciences,1997,55(1):119 - 139.

3 实验和结论

提出了一种基于方向直方图的 Adaboost 人脸检测算法,为了测试其正确性,选取了 1502 幅 320×240 图片,提取 4922 个人脸训练样本。用训练好的检测器检测独立于训练集的测试集合,共计 196 幅图片,其中包含人脸 368 个。并将像素点的梯度方向划分为 32 个子集。图 2 给出了一个灰度样本(包括其原图水平梯度图、垂直梯度图、梯度幅值图)。图 3 是梯度方向

(上接第 4 页)

统、文件系统和图形构件。文中探讨了作为 CAR 构件技术的精髓之一的“和欣”命名服务的机制及其特点,并用一个示例来说明如何使用“和欣”操作系统的命名服务。对于正确理解“和欣”操作系统的命名机制并利用“和欣”的命名机制简化在“和欣”上的编程,正确理解“和欣”系统的 CAR 构件运行环境和灵活内核机制,具有一定的价值和指导意义。同时“和欣”的命名服务机制弥补了微软 COM 的名字对象机制的不足,对于其它系统的命名服务的设计也具有指导和借鉴作用。

参考文献:

- [1] Koretide CAR'S Manual[M/CD]. 2005 - 06. <http://www.koretide.com.cn>,2004/2005.
- [2] 科泰世纪. 和欣 2.0 资料大全[EB/OL]. 2005 - 12. <http://www.koretide.com.cn>.
- [3] OMG. CORBA 服务[M]. 韦乐平译. 北京:电子工业出版社,2002.
- [4] Chen Rong. The Application of Middleware Technology in Embedded OS[C]//Workshop on Embedded System, In Conjunction with the ICYCS(6th). Hangzhou:[s.n.],2001:1 - 3.
- [5] BOX D. COM 本质论[M]. 潘爱民译. 北京:中国电力出版社,2001.
- [6] 潘爱民. COM 原理与应用[M]. 北京:清华大学出版社,1999:25 - 32.