

基于 Elastos 的构件化驱动编程模型的研究

杨向科, 陈 榕

(同济大学 基础软件工程中心, 上海 200092)

摘 要:传统操作系统中驱动程序的模型, 尤其类似 UNIX 系统中设备驱动程序往往是同文件系统相关联的, 这导致驱动程序用户接口不够灵活, 而新的构件化的驱动程序模型将增加这种灵活性。讨论基于 CAR 构件技术及 Elastos 平台来构建构件化驱动程序的方法。上海科泰世纪科技有限公司实现了 CAR(Component Assembly Runtime)构件技术以及为其提供运行时支撑的 Elastos 构件运行平台。CAR 构件技术为驱动程序的构件化、驱动程序的加载和卸载提供了技术支持。

关键词:Elastos; CAR; 构件技术; 驱动; 设备驱动模型

中图分类号: TP311

文献标识码: A

文章编号: 1673-629X(2008)12-0025-03

Research of Component Device Driver Model Based on Elastos

YANG Xiang-ke, CHEN Rui-g

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

Abstract: In traditional operating system especially in UNIX, the device driver usually depends on the file system. This results in the flexibility of user interface. A new component-based device driver model would improve the disadvantage. In this paper, how to build a component-based device driver on CAR(component assembly runtime) technology and Elastos platform is discussed. Elastos platform, which based on CAR component technology, is designed and implemented by Shanghai Koretide Co. CAR offers technic support for pervasive computing.

Key words: Elastos; CAR; component technique; driver; device driver model

0 引 言

随着嵌入式应用的广泛普及, 设备多样化, 都使得对于设备驱动程序的研究显得更为重要, 模块化、便于升级、稳定性成为设备驱动程序必要的特性, 为了适应新的需求, 有必要改造驱动程序模型, 建立新的驱动程序模型。

嵌入式操作系统 Elastos 是科泰世纪公司自行研制开发的一个基于构件化软件模型的系统。除微内核中最底层的控制部分外, 所有系统功能都是以构件接口的形式提供。

构件化驱动程序设计方案是使用 CAR 构件技术封装驱动程序, 每个驱动程序都是一个模块化非常强的构件。使用构件技术封装驱动程序使得驱动程序可

继承构件的大部分优点, 而且也给驱动程序带来了新的特性。

1 Elastos 操作系统简介

Elastos 操作系统是 32 位嵌入式操作系统, 它基于微内核, 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; Elastos 是一个完全面向构件技术的操作系统, 操作系统提供的功能模块全部基于 CAR(Component Assembly Runtime)构件技术, 因此是可拆卸的构件, 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构件^[1]。

Elastos 操作系统的最大特点就是:

1) 全面面向构件技术, 在操作系统层提供了对构件运行环境的支持;

2) 用构件技术实现了“灵活”的操作系统。这是 Elastos 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。

构件化的操作系统可以为嵌入式系统开发带来以下好处: 可以动态加载构件; 随时和动态地实现软件升级; 灵活的模块化结构, 便于移植和剪裁; 跨操作系统

收稿日期: 2008-03-28

基金项目: 国家 863 计划资助项目(2001AA113400); 国家移动通信产品研究开发专项项目(财政部(财建[2005]182 号), 信息产业部(信部请函[2005]297 号))

作者简介: 杨向科(1982-), 男, 河南许昌人, 硕士研究生, 研究方向为嵌入式操作系统、系统软件支撑技术; 陈 榕, 博士生导师, 教授, 研究方向为嵌入式系统、构件技术。

平台兼容,降低软件移植的风险。

2 CAR 编程与构件模型简介

2.1 CAR 的含义

CAR,即 Component Assembly Runtime,是在运行时对软件构件进行组装并最终完成预计功能的一种软件技术^[2]。

在 Elastos 运行环境中,Component Assembly 包含了两层含义:

(1)软件零件,特指“目标代码单元”。在 CAR 编程规范中就是 DLL;

(2)软件部件,是软件零件的集合。一般是个“半成品”,通过 XML 或脚本包装成为“产品”,也可以直接是个“产品”。软件部件不但包含一个或一组 DLL,还包含了数字签名、下载压缩包、应用描述文件、元数据信息等。类似于 JAVA 里面的 JAR 文件、Windows 里面的 CAB 文件。

2.2 CAR 的技术内涵

CAR 是一种基于构件的软件运行支持技术。它对构件的运行支持能力直接决定所支持的构件的编写方法、结构设计,甚至算法选择。CAR 支持满足“故障独立性”的运行环境。即某个部件失效不会引起其它部件的失效,这是硬件系统可靠性的基本特性。CAR 通过这种环境所提供的构件动态组装,对外完成预计的计算任务^[3]。

CAR 也是一种构件化的开发语言,它只负责框架部分描述,具体的实现逻辑由 C/C++ 等编程语言实现。CAR 所描述的框架部分以元数据的形式存在于构件的发行格式中,元数据通过反射(reflection)机制参与构件组装计算^[4]。框架是将具体的应用逻辑通过类似于微软 COM^[5]构件的方式(计数管理、接口查询、构件聚合)隐藏起来,并把框架自己暴露在外的一种最终运行封装。它提供了构件的标准,二进制构件可以被不同的应用程序使用,使软件构件真正能够成为“零件”,从而提高软件生产效率。

3 Elastos 构件化驱动编程模型

Elastos 是一个完全面向构件技术的操作系统。操作系统提供的功能模块全部基于 CAR 构件技术,这也为构件化驱动程序设计提供了技术平台。构件化驱动程序设计方案是使用 CAR 构件技术封装驱动程序,每个驱动程序都是一个模块化非常强的构件。

3.1 Elastos 体系结构简介

如图 1 所示,Elastos(Elastos)操作系统主要包含以下几个模块:板卡驱动程序;设备驱动程序;内核;系统

平台层;其它应用级服务;DDK APIs。

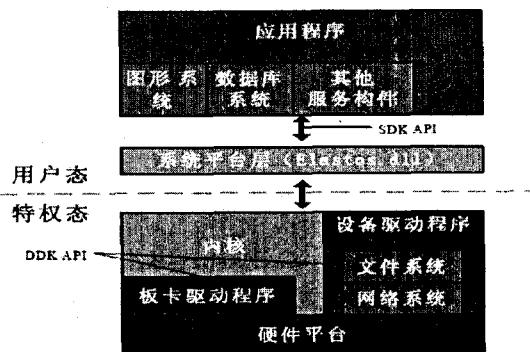


图 1 Elastos 体系结构图

3.2 设备描述信息与配置

Elastos 的内核以一个设备名(宽字符串型)加上一个设备号(无符号整型)来唯一标识一个特定的设备实例。这里所说的“设备”,可以是实际存在的硬件设备实体,也可以是虚拟的设备(伪设备)。通常情况下,设备名标识设备的类型,而设备号标识此类设备的一个实例。例如:系统中存在有两个串口设备,设备名可以为“com”,而设备号为 1 和 2。那么设备名加设备号“1”就唯一表示了第一个串口设备;而设备名“com”加设备号“2”就唯一表示了第二个串口设备^[6]。

所有设备名在系统中共享同一个名字空间,所以必须是全局唯一的。一个设备名只能对应到唯一一个驱动程序,系统将以设备名来匹配相应的驱动程序。对系统中已经存在的设备,它们的信息可以静态地放入一张全局设备配置表中。内核为这张全局设备配置表维护一个以 struct DeviceConfig 为元素类型的数组 s_deviceConfigs 加上一个 uint_t 类型的全局变量 uNumberOfDeviceConfigs 来指明该数组的实际长度。可以通过标准内核功能接口 DzRegisterDevice() 函数动态地向系统注册设备信息。注销设备信息则是通过标准内核功能接口 DzUnregisterDevice()。

3.3 驱动程序接口的实现与配置

Elastos 设备驱动程序以静态方式被链入内核映像文件成为内核的一部分。与内核的其他部分一样,设备驱动程序代码在处理器特权态下执行。除了这些特征之外,Elastos 设备驱动程序本质上就是一类存在于内核之中的特殊 CAR 构件。它们都必须通过暴露出一个统一的 CAR 接口 IDriver 来提供服务。

所有的内核驱动程序都提供了一致的 IDriver 接口,但是却以各自特定于设备的方式来解释 IDriver 接口的精确语义。通常,IDriver 接口的 Read() 方法用于从设备中读入数据;Write() 方法用于向设备写数据;而 Control() 方法则用于向设备发送命令、获取状态信息等等。

下面就以鼠标驱动开发为例,来介绍一下构件化驱动编程模型的具体开发方法,采用 C++ 语言。

1) 定义一个驱动 C++ 类,继承自 IDriver,其定义如图 2 所示。

```
class Driver : public IDriver {
public:
    CARAPI QueryInterface(
        /* [in] */ REFIID riid,
        /* [out] */ void **ppv);

    CARAPI_(ulong_t) AddRef(void);

    CARAPI_(ulong_t) Release(void);
    ...
    virtual void Dispose() = 0;
    ...
};
```

图 2 一个驱动 C++ 类

IDriver 接口的 Read()、Write()、Control() 方法也必须由派生的驱动类来实现。就上例而言,鼠标驱动类的定义可以如图 3 所示。

```
class Mouse : public Driver {
public:
    CARAPI Read(
        /* [in] */ UINT64 u64Offset,
        /* [in] */ UINT uNumberOfBytesToRead,
        /* [out] */ ByteArray ebbData,
        /* [out] */ IEvent ** ppCompletionEvent);
    CARAPI Write(
        /* [in] */ UINT64 u64Offset,
        /* [in] */ ByteArray ebbData,
        /* [out] */ UINT * puNumberOfBytesWritten,
        /* [out] */ IEvent ** ppCompletionEvent);
    CARAPI Control(
        /* [in] */ INT nControlCode,
        /* [in] */ ByteArray ebbInData,
        /* [out] */ ByteArray ebbOutData,
        /* [out] */ IEvent ** ppCompletionEvent);
    virtual void Dispose();
public:
    Mouse(...);
    ~Mouse();
    ECODE InitHw(); // 检测并初始化硬件设备
public:
    ...
};
```

图 3 鼠标驱动类的定义

2) 定义并实现 IDriver 接口的 Read()、Write()、Control() 方法。

3) 为驱动程序实现一个构造函数(工厂方法),负责进行驱动构件实例的创建和初始化。

所有的驱动实例构造函数在系统中共享同一个名字空间,所以必须拥有全局唯一的名字。建议的命名规则为:“CreateDevice”,其中“Device”部分应被替换成具体驱动程序所对应的设备名。就上例而言,可命名

为:“CreateMouse”。

创建函数的具体实现也与该驱动程序所要求的创建模式有关。一般情况下一个驱动程序会有多个实例,此时可以直接运用 C++ 的 new 操作符从内核堆中分配驱动构件实例。就上例而言,可以实现如图 4 所示。

```
EXTERN IDriver * CDECL CreateMouse(
    uint_t uDeviceNo, void *pvParameter)
{
    ...
    Mouse *pMouse = new Mouse (...);
    if (NULL == pMouse) return NULL; // 内存不足
    ...
    ECODE ec = pMouse->InitHw();
    if (FAILED(ec)) { // 检测不到硬件设备或初始化硬件设备出错
        delete pMouse;
        return NULL;
    }
    ...
    pMouse->AddRef();
    return pMouse; // 创建驱动构件实例成功
}
```

图 4 创建驱动构件实例

相应地,该驱动类的 Dispose() 方法中应该运用 C++ 的 delete 操作符来删除对象,以释放内存。就上例而言,可以实现如图 5 所示。

```
void Mouse::Dispose()
{
    // 关闭硬件设备,注销 ISR
    ...
    delete this;
}
```

图 5 删除对象

4) 使用设备驱动程序的服务。

```
...
IDriver *pDriver;
ECODE ec = EzFindService(
    EZCSTR("device:mouse"), (IUnknown **)&pDriver);
if (FAILED(ec)) {
    // 出错处理
}
...
ec = pDriver->Read(...); // 读取鼠标输入
if (FAILED(ec)) {
    // 出错处理
}
...
// 通讯完毕,释放接口
pDriver->Release();
...
```

图 6 鼠标驱动客户程序

当系统成功创建出了一个驱动实例,就会通过 Elastos Service 机制注册此驱动实例的 IDriver 接口。客

(下转第 31 页)

客观上评价匹配效果的好坏,引入三个评价指标^[8]:

- * $B_{\bar{O}}$ — 非遮挡区匹配误差大于 1 的百分比。
- * $B_{\bar{T}}$ — 非纹理区匹配误差大于 1 的百分比。
- * B_D — 接近不连续区匹配误差大于 1 的百分比。

$B_{\bar{O}}$ 、 $B_{\bar{T}}$ 、 B_D 的表达式分别为:

$$B_{\bar{O}} = \frac{1}{N} \sum_{s \in \bar{O}} (|d(s) - d_T(s)| > 1) \quad (15)$$

$$B_{\bar{T}} = \frac{1}{N} \sum_{s \in \bar{T}} (|d(s) - d_T(s)| > 1) \quad (16)$$

$$B_D = \frac{1}{N} \sum_{s \in D} (|d(s) - d_T(s)| > 1) \quad (17)$$

公式(15)~(17)中 N 分别是对应区域的总像素点个数, $d(s)$ 表示像素点的视差估计值, $d_T(s)$ 是像素点的实际视差值。对图 4 中的(b)、(c)、(d) 计算 $B_{\bar{O}}$ 、 $B_{\bar{T}}$ 、 B_D 得到结果如表 1 所示。

表 1 误匹配率对比

视差图	$B_{\bar{O}}(\%)$	$B_{\bar{T}}(\%)$	$B_D(\%)$
b(灰度)	2.15	0.82	15.13
c(RGB)	1.72	0.74	9.54
d(HSI)	1.69	0.76	10.32

由表 1 可以看出,利用 RGB 和 HIS 信息的算法在非遮挡区、非纹理区和不连续区的误匹配率均小于利用灰度信息的算法,特别是在不连续区,误匹配率显著降低,显示出基于颜色空间的算法在不连续区的匹配上有很大的优势。

5 结束语

对测试图片的匹配比较实验表明,利用颜色信息代替灰度信息构建匹配代价函数,可以为立体图像对的全局匹配提供更丰富的信息。相对于基于灰度的 MRF 立体匹配算法,基于颜色空间的 MRF 算法可

以得到精确度更高的匹配结果。尽管加入颜色信息降低了误匹配率,但是不连续区的误匹配率还是很大,进一步工作可以研究更加有效地处理不连续区匹配的匹配方法。

参考文献:

- [1] Cai Xuan - ping, Zhou Dong - xiang, Li Gan - hua, et al. A stereo matching algorithm based on color segments[C]//IEEE International Conference on Intelligent Robots and Systems. Edmonton, Canada: [s. n.], 2005: 3372 - 3377.
- [2] 顾 征, 苏显渝. 采用色彩相似性约束的立体匹配[J]. 光电工程, 2007, 34(1): 95 - 99.
- [3] 郭龙源, 夏永泉, 杨静宇. 一种改进的彩色图像匹配算法[J]. 计算机工程与应用, 2007, 43(27): 98 - 104.
- [4] Chambon S, Crouzil A. Color stereo matching using correlation measures[J]. Complex Systems Intelligence and Modern Technological Applications, 2004, 23(5): 520 - 525.
- [5] Sun Jian, Zheng Nan - ning, Shum Heung - yeung. Stereo matching using belief propagation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003, 25(7): 1 - 14.
- [6] Tappen M F, Freeman W T. Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters [C]//IEEE International Conference on Computer Vision. Nice, France: [s. n.], 2003: 900 - 907.
- [7] Boykov Y, Veksler O, Zabih R. Fast approximate energy minimization via graph cuts[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001, 23(11): 1222 - 1239.
- [8] Scharstein D, Szeliski R. A taxonomy and evaluation of dense two - frame stereo correspondence algorithms[J]. International Journal of Computer Vision, 2002, 47(1): 7 - 42.

(上接第 27 页)

户程序可以通过 Elastos Service 机制寻找特定的设备驱动实例提供的服务。如果能够获取到驱动程序的服务接口 IDriver, 就可以通过 IDriver 的方法来请求设备的 I/O 服务了。以“驱动程序接口的实现与配置”一文中所举的鼠标驱动为例, 客户程序可以实现如图 6 所示。

4 结束语

着重介绍了在嵌入式操作系统 elastos 平台下, 基于 CAR 构件技术的驱动程序开发模型, 它和传统的操作系统驱动程序模型有所不同, 它降低了驱动程序和系统其它部分之间的耦合程度, 而且能够更清晰地描述硬件设备, 使驱动程序更便于使用。

参考文献:

- [1] 上海科泰世纪技术有限公司. Elastos 资料大全[EB/OL]. 2006 - 10. <http://www.koretide.com.cn>.
- [2] 上海科泰世纪技术有限公司. CAR 构件技术与编程模型[EB/OL]. 2006 - 10. <http://www.koretide.com.cn>.
- [3] 上海科泰世纪技术有限公司. Elastos 技术白皮书[EB/OL]. 2006 - 10. <http://www.koretide.com.cn>.
- [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] 潘爱民. COM 原理与应用[M]. 北京: 清华大学出版社, 1999: 25 - 32.
- [6] Corbet J, Kroah - Hartman G, Rubini A. Linux Device Drivers[M]. 3rd Edition. [s. l.]: O'Reilly, 2005.