Fuxi 语言的仿真环境

1，王忠斌[1]　2，韩仁顺
（1．深圳市海创达资讯技术有限公司计算新技术实验室　深圳 518048
2．深圳市冈野科技有限公司 广东深圳 518049）

摘要：Fuxi 语言是一种面向方面、平台无关的说明性程序设计语言，Fuxi 程序可以同时在目标设备和台式电脑上执行。但桌面系统和目标设备毕竟存在着巨大的系统差异性，在桌面环境中执行的 Fuxi 程序很难反映目标设备的真实性。本文介绍一种利用 Fuxi 语言的面向方面和平台无关性开发的 Fuxi 嵌入式应用的桌面仿真环境，它能真实地模拟目标设备，为开发人员和用户提供一个有效的测试和评估环境。
关键词：嵌入式系统；中间件；仿真；说明性语言；Fuxi 语言；Fuxi 平台

An Emulation Environment for Fuxi Language

1, WANG Zhongbin　2, HAN Renshun
(1. Laboratory of Novel Computing, Hitrend Information Technologies, Inc. Shenzhen GD 518408
2. Okano (Shenzhen) Technologies, Ltd. Shenzhen GD 518049)

Abstract: Fuxi is an aspect-oriented, architecture-independent programming language, can help user to build concise embedded applications. Because of the tremendous difference between target device and desktop systems, one usually can not verify and evaluate the programs through directly running them on desktop environment. In this paper, we introduce a programmable emulation environment for Fuxi applications, which can emulate the target device in desktop environment, thus provide a good tool for program verification and evaluation.
Keywords: Embedded Systems; middleware; emulation; declarative language; Fuxi language; Fuxi Platform

1. 前言

Fuxi 程序设计语言[1]是一种面向方面、面向对象、平台无关的说明性语言。Fuxi 语言结合了函数型、逻辑型、面向对象等语言的特点，具有很强的表达力，可以用来构造简洁的应用程序。同时，Fuxi 语言在实现技术上注重表达力与效率的结合，使之具有广泛的应用前景。目前 Fuxi 语言已经在普适计算[2]、嵌入式计算[3，4]、互联网以及工业控制与测试等领域取得了应用。

由于 Fuxi 语言的平台无关性，Fuxi 语言所开发的应用程序可以同时在桌面环境和目标设备上运行。因此，Fuxi 语言为嵌入式应用提供了一个良好的 PC 仿真条件。但是，由于桌

作者简介：王忠斌，安徽来安人，ACM 会员、CCF 高级会员，高级工程师，长期从事程序设计语言、中间件、嵌入式系统、集成化开发环境等研究。

面系统和目标设备之间往往存在着巨大的系统差异性，在桌面环境中执行的 Fuxi 程序很难仿真出目标设备的状态，这就给测试和评估带来了一些困难。

为此，我们利用 Fuxi 基于元对象模型的面向方面的机制和平台无关性的特点，开发了一种能在 PC 桌面环境中，仿真目标设备的 Fuxi 仿真环境。开发人员可以在桌面系统上仿真目标设备上的运行效果，方便地对应用程序进行测试、验证和性能评估。

本文在第 2 节简单介绍 Fuxi 的嵌入式应用系统模型和编程模式；在第 3 节讨论 Fuxi 仿真环境的系统模型；第 4 节重点介绍 Fuxi 仿真器的编程方法；在第 5 节进行总结。

2. Fuxi 的嵌入式编程模式

在设计 Fuxi 的仿真环境之前，首先需要分析 Fuxi 语言的嵌入式编程模式。和 JAVA 类似，Fuxi 程序也是由一些类构成的。Fuxi 语言中所有的类都是从一个 fuxi.Object 的元对象派生出来的。图 1 是 Fuxi 语言的元对象的层次结构。

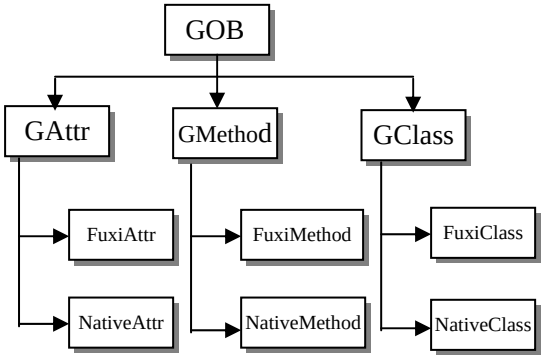

图 1. Fuxi 的元对象层次结构

Fuxi 系统中的类，可以是用 Fuxi 语言编写的 Fuxi 类，也可以是利用 Fuxi 元对象模型，采用本地代码实现的本地类(Native Class)。Fuxi 的本地类通常采用 C/C++来编程。Fuxi 类具有平台无关性，而本地类具有高执行效率。只要为 Fuxi 类提供支持的本地类在不同平台上都具有相同的调用协议，则 Fuxi 类就可以从一个平台上迁移到另一平台上执行(见图 2)。

我们把遵守公共协议的本地类库(Fuxi 的基本库)和 Fuxi 抽象机一起称为 Fuxi 平台。因此，Fuxi 平台也是一个中间件平台。

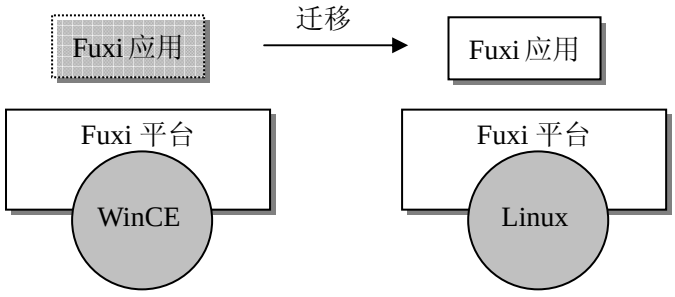

图 2. Fuxi 平台上的 Fuxi 应用的可移动性

2.1 Fuxi 应用的架构

一个可执行的 Fuxi 程序通常是从 Application 类派生出来的应用，Application 是 Fuxi

基本库提供的一个主动式本地类。当 Fuxi 抽象机创建 Fuxi 应用的实例时，将自动为其分配一个执行线程和相关的运行栈。

Application 中包含一个 Platform 的静态引用成员，我们可以通过 Application 中的方法 GetPlatform()来获取 Platform 对象的引用。

Platform 也是一个本地类，是系统平台与应用程序之间的桥梁(如图 3 所示)。我们可以通过 Platform 所提供的方法获取系统服务；同时，当系统状态发生变化时，Platform 也会向应用程序发送相应的事件，应用程序可以通过触发器方法来进行相应的处理。

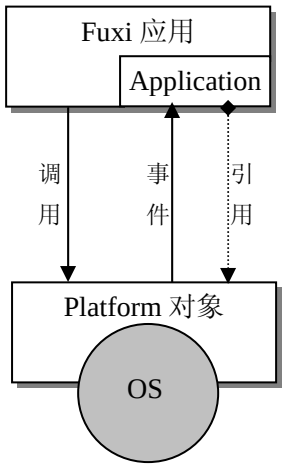

图 3. Fuxi 应用与平台之间的关系

以下是一个典型的 Fuxi 应用的程序框架。

```
import fuxi.*
……
public active class CGPSShell : Application
{
    CMainPanel     m_MainPanel ( )                    // 用户 GUI 主界面
    ……
    Time m_tmStartup = GetPlatform().GetSystemTime()
    public Activate( ) =
    {
        SetControl( m_MainPanel )
        base.Activate()
        m_MainPanel.Activate( )                       // 激活 GUI 主界面
    }
    public OnCommand( MAINITEM + 1 ) -> { … }         // GUI 菜单事件
    ……
    public OnDevRemoved( "\\存储卡" ) -> { … }        // 设备事件
    public OnDevAppeared( "\\存储卡" ) -> { … }
    ……
}
```

程序中，我们使用了 GetPlatform()函数获取 Application 中的 Platform 对象，并调用 Platform 的 GetSystemTime()函数获取系统时间；而 OnDevAppeard("\\存储卡")和 OnDevRemoved("\\存储卡")则是用于响应插拔存储卡的设备事件。

Platform 类是同相应的目标平台紧密关联的，虽然 Fuxi 程序是平台无关的，可以在不同的平台环境中执行，但是由于各平台环境向 Fuxi 应用程序所提供的 Platform 对象的差异性，同一 Fuxi 程序在不同的平台中将可能表现出不同的行为。因此，把为嵌入式设备开发的 Fuxi 应用，在桌面环境中直接运行，常常不能反映目标设备的真实性。

3. Fuxi 的仿真模型

所谓 Fuxi 仿真环境，就是通过一定的技术手段，使得桌面环境(如 Windows)也能向 Fuxi 应用程序提供一个具有和目标设备相同或近似行为的 Platform 对象，并提供一个用户界面或接口，使得用户可操控该 Platform 对象，使之能够发生一些目标设备中可能会出现的事件。这样，开发人员或用户就可以利用该仿真环境来验证应用程序的正确性，或对程序的功能进行评估。

图 4. 是 Fuxi 仿真环境工作的原理图。

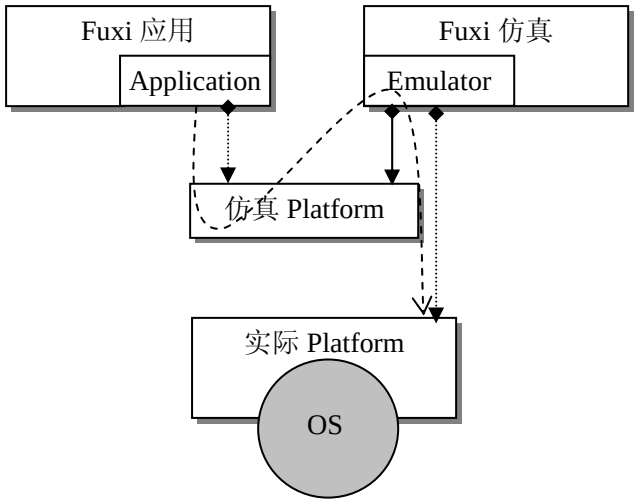

图 4. Fuxi 仿真环境的工作原理

Emulator 是 Fuxi 仿真包 emul 中的一个主动式本地类，它和 Fuxi 基本库中的 Application 类似。在系统创建 Fuxi 仿真器时，Emulator 将根据仿真程序的要求创建一个仿真 Platform 对象，同时将该仿真对象注册到 Fuxi 环境的注册表中。在 Application 初始化中，Application 将从 Fuxi 环境中获取该仿真 Platform 的引用，而不再是实际系统的 Platform 对象。

这样，Fuxi 应用与实际 Platform 之间的之间联系就被仿真 Platform 所阻断，变成了经过仿真 Platform 和 Emulator 桥接的间接联系。Fuxi 应用对实际 Platform 的调用，将由 Emulator 来仿真完成，而实际 Platform 向 Fuxi 应用发送的事件也经过了 Emulator 的过滤。同时，Emulator 还可以在用户的控制下，向 Fuxi 应用发送一些实际 Platform 所不存在或未发生过的事件。

4. 仿真器编程

在开发 Fuxi 应用的同时，可以根据目标设备的规格说明，编写一个目标设备的仿真程序。Fuxi 语言的仿真包 emul 为我们提供了开发设备仿真程序的工具。

一个 Fuxi 仿真程序通常包括以下部分：

1. 一些具有特定功能的设备对象，如 Keypad 等；
2. 一个用户操作界面；
3. 包含上述成员的 Emulator 聚集类。

4.1 仿真程序的框架

每个 Fuxi 仿真程序都是从 emul 包中的 Emulator 类派生出来的。仿真包还定义了一个抽象类 Device，Emulator 是一些具有特定功能的 Device 对象的聚集(Aggregation)。Emul 包中已经为我们定义了一些常见的设备，如 Keypad、TouchPad 等。用户也可以根据需要利用 Device 类来定义自己的设备。

此外，emul 包中还包括一些设备控件，如 Screen。用户可以利用这些控件定义仿真器的用户操作界面。

以下是一个 Fuxi 仿真程序的框架。

```
import fuxi.*
import emul.*
……
public active class GPSEmulator : Emulator
{
    EmuPanel m_EmulPanel( )              //用户操作界面
    Keypad m_devKeypad()                 //Keypad 设备
    ……
    public Activate( ) =
    {
        SetControl( m_EmulPanel )
        base.Activate( )
        m_EmulPanel.Activate()           //激活仿真面板
    }
    public OnCommand( 1001 ) ->          //用户菜单命令
    {
        m_devKeypad.pressKey( 'A' )
        delay( 20 )
        m_devKeypad.releaseKey( 'A' )
    }
    ……
}
```

上面的仿真程序包含一个用户界面 m_EmulPanel，它可以采用 FuxIDE 中的资源编辑器

来进行编辑，可以将其设计成和真实设备一样的外观(见图 5)；还包括一个 Keypad 设备，在用户操作界面中采用图形按钮来模拟。

OnCommand()触发器用来响应按钮消息，当点击操作界面中的按钮后，使用 Keypad 类中的 pressKey()和 releaseKey()方法，通过仿真平台向应用程序发送一个‘A’键消息。这样，就可以验证应用程序对目标设备上的按键处理功能的正确性。

4.2 仿真环境的执行过程

Emulator 是一个主动类，其实例是一些具有独立线程的主动式对象。在仿真执行过程中，仿真程序和应用程序并行执行。Application 和 Emulator 之间的桥梁就是图 4 中的“仿真 Platform”对象。

Fuxi 抽象机的执行参数中包括一个 TARGET 选项，如下：

fuxi <模块名> <参数表> [-TARGET:<模块名>] [-DEBUG]

当 TARGET 选项指定为某个 Fuxi 程序(Fuxi 仿真程序当然也是一个 Fuxi 程序)后，抽象机创建完应用对象后，在启动应用线程之前将启动“目标程序”。此时，Emulator 将其创建的“仿真 Platform”对象注册到 fuxi 环境中，在 Application 对象激活时，从 fuxi 环境中获取的平台对象就是 Emulator 所提供的“仿真 Platform”了。当用户终止仿真程序时，“仿真 Platform”向应用程序发出停机事件，结束应用程序的执行。

图 5. 一个仿真器的操作界面

5. 结论

Fuxi 仿真包提供了一个目标设备的可编程仿真环境。软件开发人员可根据目标设备的规格说明，便捷地编写一个目标设备的 Fuxi 仿真器，并将开发的应用程序定向到这个仿真器上执行。可以通过仿真器向应用程序发送一些仿真的信号或事件，用于测试、验证应用程序的正确性，评估应用程序的性能。

参考文献

[1] 海创达，“Fuxi 程序设计语言规范”[OL]，见 http://www.fuxi.org，2006.4

[2] 王忠斌，“Fuxi：一个普适计算的语言及平台”[C]，《第二届全国普适计算学术会议论文集》2006.10 杭州

[3] 王忠斌、黄继东，“Fuxi：一种敏捷的嵌入式开发环境”[C]，《全国第 6 届嵌入式系统学术年会论文集》下册，2006.11 西安，pp206-210

[4] Wang, Z., “Fuxi: An agile development environment for embedded systems”[C], Proc. IEEE COMPSAC Beijing. 2007.7, pp631-632