基于软件复用的嵌入式操作系统的定制

李向蔚，桑 楠，熊光泽

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】嵌入式系统的专用和资源约束等特性要求嵌入式操作系统必须是可定制的。以软件模块化复用设计为基础的操作系统剪裁配置技术为应用开发者定制系统提供了有力的支持。为了降低配置方式与工具的多样性所导致的定制操作系统的难度，该文基于模块的抽象关系提出了一种统一的定制过程模型，并在此基础上实现了一个嵌入式Linux的配置剪裁器。

关 键 词 嵌入式开发； 嵌入式系统； 操作系统定制； 裁减

中图分类号 TP316.2 文献标识码 A

Embedded Operating System Customizing with Software Reusing

LI Xiang-wei，SANG Nan，XIONG Guang-ze

(School of Computer Science and Engineering, University of Electronic Science and Technology of China Chengdu 610054)

Abstract The customization of embedded system as well as its resources constraint requires that the embedded operating system must be configurable. The design of reusable modularized software is the base of operating system tailoring and configuration. Based on abstract relation among different components, this paper proposes an uniform customizable procedure model in order to reduce the difficulty of configuration. With the help of this model, a configure-tailor on embedded Linux is realized.

Keywords embedded development; embedded system; operating system customizing; tailoring

嵌入式系统开发已经成为计算机工业最热门的领域之一，其应用渗透到信息家电、工业控制、通信与电子设备、人工智能设备等诸多领域。嵌入式操作系统的引入大大提高了嵌入式系统的功能，方便了嵌入式应用软件的设计，但同时也占用了宝贵的嵌入式资源[1]。嵌入式应用系统配置差别较大，大部分外设驱动软件也没有标准化，这就要求在开发环境中对操作系统进行剪裁和扩展，使之和应用程序绑定在一起生成一个可运行在目标环境下的高效应用系统。这是降低系统硬件成本，减少系统资源消耗，提高系统灵活性的关键。

软件复用是解决软件危机，提高软件生产效率和质量的现实可行的途径。软件复用是指重复使用“为了复用目的而设计的软件”的过程，而基于构件或模块的软件开发是软件复用的主要形式[2]。嵌入式操作系统大多采用组件化、模块化的设计思想，以搭积木的方式通过互连构造软件[3]，因而是可配置的。但是由于操作系统的多样性，不同操作系统提供的配置方式迥异且繁简不一[4-5]。而由于硬件平台的多样性，即使是相同的操作系统，其应用配置也有差别。结果，应用程序开发者必须熟悉不同的硬件平台和操作系统才能进行有效的应用开发，增加了应用开发的难度。

为解决上述问题，本文将软件复用技术用于嵌入式操作系统的定制过程，提出了一个统一的嵌入式操作系统定制过程模型OSTAILOR。该模型已应用于嵌入式Linux的定制过程中。

1 嵌入式操作系统定制过程模型

本文以抽象的操作系统模块作为操作系统配置的基本单位，构造了嵌入式操作系统定制过程模型OSTAILOR。

1.1 模型概述

考虑到各类操作系统的差异性，OSTAILOR要求对不同操作系统的物理模块进行功能抽象，使之对应于定制过程模型所用逻辑模块，从而在逻辑上对不同的操作系统物理模块保持透明，这是实现操作系统定制通用性的前提。按照并集原则{抽象OS功能集}={Vxworks功能集}∪{Linux功能

收稿日期：2005－02－11
基金项目：国家863高技术研究发展计划资助项目(2003AA1Z2210)
作者简介：李向薇(1981－)，女，硕士，主要从事嵌入式实时系统应用与开发技术方面的研究.

集}∪{CRTOS功能集}∪…，模块实体被抽象封装成对OSTAILOR可用的抽象模块，它包含两个关键属性：(1) 抽象出来的模块名：用于确定此模块具有的基本功能。(2) 模块层次：用于标识隶属于不同层次的模块所具有的不同粒度。低层次、细粒度的模块可以组合形成更高层次、更大粒度的模块。每个级别的模块都可以在操作系统定制过程中使用。通常，模块粒度越大，其包含的功能越多，可配置性就越低。模块粒度越小，配置复杂度越高[6]。为了兼顾配置的灵活性和方便性，并将其中涉及到的操作系统特殊性降到最小，本文将模块结构表达为如图1所示的抽象的模块关系。

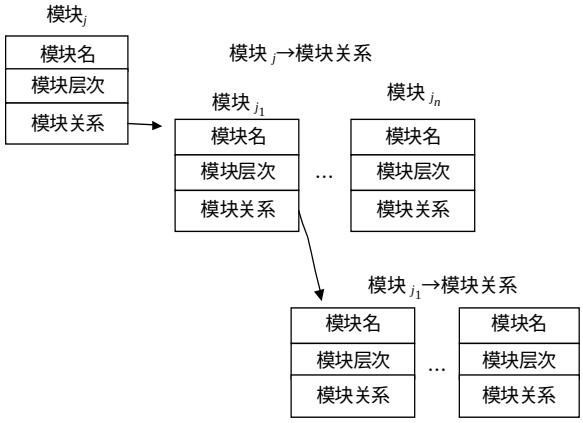

图1 模块的抽象关系图

模块的抽象关系可形式化描述如下：

$$\text{module}=\{\text{moduleName}, \text{moduleLevel}, \text{elationship}\}$$

$$\text{moduleLevel}=(1, 2, \cdots, n)$$

$$\text{Ralationship}(\text{module}_j)=\{\text{module}_{i_1}, \text{module}_{i2}, \cdots, \text{module}_{i_n}) \mid \text{LevelOfModule}(\text{module}_{i_x})= \text{LevelOfModule}(\text{module}_j)+1\}$$

$$\text{LevelOfModule}(\text{module}_j)=\{\text{module->moduleLevel} \mid \text{module}=\text{module}_j\}$$

式中 用module_j表示任一抽象模块j；moduleName、moduleLevel、Relationship分别表示抽象模块的三个属性域：模块名、模块层次和模块关系，如module_j->Relationship表示模块j的模块关系，module_{j_k}(k为任一整数)表示module_j->Relationship下包含的一个模块k。

这些抽象模块以属性文件的形式存放在构件库，供配置选用。以模块抽象关系为基础的定制模型的结构如图2所示，它包括操作系统抽象层和操作系统适配层。

操作系统抽象层是对用户和应用可见的接口层，它通过模块抽象，屏蔽不同操作系统表现出来的特性。用户使用这层提供的接口，通过对不同粒度的抽象模块进行选择来进行操作系统配置，生成一个对用户和应用可见的虚拟操作系统，而无需考虑所定制的操作系统类型。

操作系统适配层包括模块解析和特定操作系统映射两部分。采用创建模块层次关系图的方法，模块解析将操作系统抽象层配置的所有复合的大粒度抽象模块解析成OSTAILOR中粒度最小的模块集。该模块集是特定操作系统映射部分接受的唯一输入形式。特定操作系统映射受到输入激励而启动，它把用户定制的操作系统类型作为多路开关来选择不同的操作系统映射算法，把接收到的模块映射到具体的操作系统实体。操作系统适配层通过其特定的处理流程，将上层用户可见的虚拟操作系统和下层真实的操作系统联系起来。

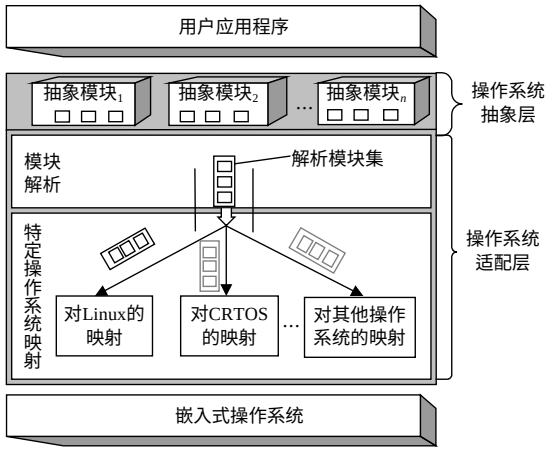

图2 OSTAILOR结构

1.2 基本工作原理

1.2.1 抽象模块的组织形式

依据模块抽象形式中的域module->Relationship，在模块容器构件库中将抽象模块组织为如图3所示的层次，若干个下层模块通过不同的组合封装出上层模块，形成不同的模块粒度和级别。

最下层是原子级模块，它抽象了操作系统的基本功能，包括任务(Task)、I/O驱动(Driver)、定时器(Timer)、信号量(Semaphore)、消息队列(Queue)、事件(Event Group)、邮箱(MailBox)、管道(Pipe)等内核管理原语，作为操作系统配置必选项存在的微内核及其他如存储管理、进程控制、文件系统管理、网络协议等操作系统组件功能。原子级模块是不可再分的基本模块，是配置其他层次模块的基础。此外，它还可以包含一些基本的应用子功能抽象，如视频压缩、图像编码、图像解码、无线上网等。

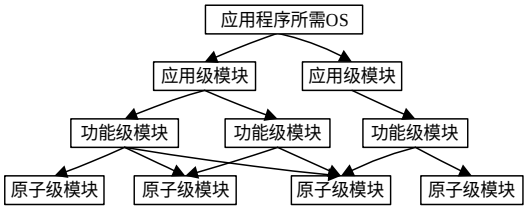

图3　模块分层组织形式

原子级模块是粒度最细的一层，和操作系统实体联系最紧密，这些实体以目标文件或库文件形式存在。在模块配置过程结束以后，通过该层映射到不同的物理实体，产生不同格式的配置文件，达到操作系统定制的目的。原子级模块是只完成特定功能的一个子功能，不能独立地使用。因此为了完成某一特定功能，就需把原子级模块组合在一起，形成大粒度的功能级模块。如配置无线上网的功能模块，除了要选择无线上网本身的原子级模块，还需要选择操作系统中支持上网功能的原子级模块，如TCP/IP协议栈。

每一个应用级模块都是一个功能级模块集，它是粒度最大的一层模块。应用程序开发者为了开发特定的应用程序，可直接选择应用级模块来配置操作系统以避免选择小粒度模块的繁琐。

模块化的分层结构为开发者提供了不同的模块粒度，使其可以根据需要，选择不同层次的模块，增强定制过程的灵活性。当应用需求改变时，不必重新经历一次完整的操作系统定制过程，而可以通过替换相关模块的方式，在保持其他部分不变的情况下，实现定制，减少应用开发时间。

1.2.2　定制流程

OSTAILOR的定制流程如图4所示，用户只选择模型的输入，包括所定制的操作系统类型和操作系统抽象模块。OSTAILOR首先接收用户输入进行模块配置，经过模块解析处理，输出相关的配置文件，然后生成所定制的系统。具体定制流程说明如下：用户浏览器查询构件库，并对用户显示出可选模块，然后接收用户对模块的选择，将选择结果送至文件解析器。再从构件库获得对应模块的属性文件，通过解析，将模块解析结果生成模块结构映射关系图。模块结构关系图中的模块均以抽象逻辑形式存在，可以通过查找构件库得到它以文件形式存在的物理实体信息。根据模块结构映射关系图和每一个模块的物理实体信息，生成对应的Makefile或其他一些相关配置文件，把它提供给对应的Make工具(编译器、链接器等)，最后生成系统的可执行文件。

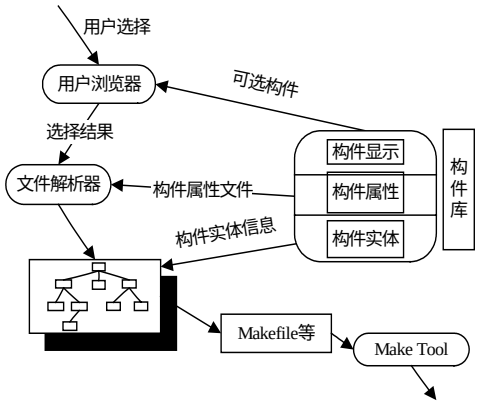

图4　操作系统定制流程

1.2.3　模块层次的解析

模块解析和模块配置过程相反。模块配置是从最底层开始，逐层构建出高层模块；而模块解析则从最上层开始，层层向下，直到寻找到最下层的映射关系，同时构造出对应的关系图。基于对模块的封装，每层模块只关心和它紧密联系的下层模块的映射关系，即它是和下层的模块组合而成的。为此，在配置生成每一个模块时，将它和下层的映射关系保存在构件库中。

逐层解析时，把模块名作为关系图的顶层结点，在构件库中查找它和下层模块的映射关系。如果关系图中不存在对应于这些下层模块的结点，就在关系图中添加新的结点。根据映射关系，添加图中的拓扑关系，即相关结点的出度和入度。对于关系图中的每一个新结点，重复上述过程，直至最终结点不再映射到下层模块(该结点即对应原子级模块)，则构造出一个完整的关系图。可以说，关系图生成的过程也就是整个解析的过程。

1.2.4　操作系统映射和配置文件生成

在关系图中，把原子级模块对应的结点映射到不同的操作系统实体，再根据不同编译器链接器能识别的规则，生成不同的makefile文件、config文件和资源文件，这些配置文件的生成标志着操作系统定制过程的完成。

2　嵌入式Linux的配置剪裁器的实现

采用上述定制过程的通用模型，实现了一个嵌入式Linux的配置剪裁器，并作为一个工具集成到基于嵌入式Linux的PDA手机开发平台中。配置剪裁器使用构件库管理器作为其可选抽象模块的容器，而其输出的配置文件则提供给开发平台项目管理器，由后者提交给编译器和链接器进行操作系统生成。

三个工具与操作系统定制过程相关的功能说明如下：

1) 配置剪裁器：提供浏览模块、配置模块、定制 OS 等功能：(1) 显示从构件库获得的已有模块信息；(2) 根据用户的选择，形成新模块，并将模块信息以属性文件的形式存入构件库，供下次定制时作为已有模块使用；(3) 定制特定应用所需的操作系统：用户可以选择构件库中已存在的不同粒度的模块进行配置，由配置剪裁器生成对应的配置文件，以约定文件路径的方式传给项目管理器。

2) 构件库管理器：为配置剪裁器中对模块的层次解析和操作系统映射提供支持。它保存了供配置剪裁器查询和使用的操作系统抽象模块信息：(1) 可以配置的模块名；(2) 每个模块对它下一层的映射信息；(3) 原子模块对不同操作系统实体的映射信息。

3) 项目管理器：管理应用程序的开发，包括向配置剪裁器提供应用开发者编制的应用程序的源文件路径信息，以便生成 Makefile 文件。嵌入式软件的最大特点是应用程序不是独立存在的，它最终和操作系统绑定在一个二进制目标代码中并下载到目标机。所以，配置剪裁器需把从项目管理器获得的源文件路径信息写入对应的 Makefile 文件中。

配置剪裁器工作步骤说明如下：

(1) 按照用户选择的应用级模块生成中间文件并将其路径保存在构件库管理器中。

(2) 当项目管理器提出获取操作系统配置文件的请求时，查找构件库，得到保存的中间文件路径。根据此路径和文件名，打开文件进行解析，获得其所包含的应用级模块的个数和模块名，生成关系图的框架。

(3) 对于关系图中的每一个应用级模块，查找构件库，得到其属性文件名和路径，解析文件得到对应的功能级模块，将功能级模块和对应拓扑关系添加到关系图中。

(4) 以此类推，直到找到对应的原子级模块，最终生成一个完整的关系图。

(5) 最后，根据关系图查找构件库，把原子级模块映射到定制的操作系统实体上。系统利用 Linux 自带配置系统，通过修改 Linux 的源码系统中已有的 Makefile 和 kconfig 文件，来编译内核和其他操作系统组件，生成用户定制的操作系统。因此原子级构件对应于特定操作系统的实体信息就是 kconfig 文件中需添加的配置项。

3 结 束 语

本文通过对操作系统定制技术进行研究，抽象出操作系统定制过程通用模型 OSTAILOR，并结合某项目实现了一个嵌入式 Linux 的配置剪裁器，验证了使用此模型来实现嵌入式操作系统定制过程的可行性，把应用开发者从手工修改嵌入式 Linux 各种配置文件的繁琐中解放出来。OSTAILOR 实现了配置过程与操作系统的无关性，应用开发过程的操作系统无关性，则是本课题进一步研究的方向。

参 考 文 献

[1] 魏 忠, 蔡 勇, 雷红卫. 嵌入式开发详解[M]. 北京: 电子工业出版社, 2003.
[2] 张 路, 谢 冰, 梅 宏, 等. 基于构件的软件配置管理技术研究[J]. 北京: 电子学报, 2001, 29(2): 266-268.
[3] STALLINGS W. Operating systems: Internals and design principles[M]. 北京: 电子工业出版社, 2001.
[4] 孔祥营, 柏桂枝. 嵌入式实时操作系统VxWorks及其开发环境Tornado[M]. 北京: 中国电力出版社, 2001.
[5] 北京科银京成技术有限公司. Lambda 2.1技术白皮书[Z]. 2002.
[6] CHIVUKULA R P, BÖKE C, RAMMIG F J. Customizing the configuration process of an operating system using hierarchy and clustering[C]// Object-Oriented Real-Time Distributed Computing, 2002. (ISORC 2002). Washington, DC, USA: Proceedings Fifth IEEE International Symposium on, 2002.

编 辑 漆 蓉