TURING 图灵程序设计丛书 Linux/UNIX系列

Apress

Pro Linux Embedded Systems

Linux嵌入式系统高级程序设计

[美] Gene Sally 著
郭旭 译

- Linux嵌入式系统开发最新动态
- 以项目方式揭示Linux嵌入式开发的全过程
- 嵌入式系统开发人员必备

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 Linux/UNIX系列

Pro Linux Embedded Systems

Linux嵌入式系统高级程序设计

[美] Gene Sally 著
郭旭 译

人民邮电出版社
北京

图书在版编目（CIP）数据

Linux嵌入式系统高级程序设计 / (美) 萨莉(Sally,G.) 著 ; 郭旭译. -- 北京 : 人民邮电出版社, 2010.11
(图灵程序设计丛书)
书名原文: Pro Linux Embedded Systems
ISBN 978-7-115-23937-2

Ⅰ. ①L… Ⅱ. ①萨… ②郭… Ⅲ. ①Linux操作系统－程序设计 Ⅳ. ①TP316.89

中国版本图书馆CIP数据核字(2010)第182497号

内 容 提 要

本书共18章，内容包括如何开发嵌入式Linux系统、移植Linux及其最佳实践。深入浅出地剖析了嵌入式Linux项目，讲述了如何创建嵌入式Linux开发环境、配置和联编嵌入式Linux内核，为嵌入式系统配置和联编开源项目，最小化资源利用和启动时间的方法，以及联编项目的可用开源资源。

本书面向理解基本软件开发理念的嵌入式系统开发人员。

图灵程序设计丛书

Linux嵌入式系统高级程序设计

◆ 著 [美] Gene Sally
译 郭 旭
责任编辑 杨海玲
执行编辑 毛倩倩 李 静

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：21.75
字数：541千字 2010年11月第1版
印数：1－3 000册 2010年11月北京第1次印刷

著作权合同登记号 图字：01-2010-4740号

ISBN 978-7-115-23937-2

定价：65.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223

反盗版热线：(010)67171154

版 权 声 明

致　谢

首先，我感谢仁慈的主给予我时间、幸运、好奇心、精力、才智和健康。我希望本书的写作能够使这些天赋物尽其用。在本书写作期间，我的家人给予了我非常大的支持。没有妻儿的支持我无法完成本书：你们对我的帮助，远超你们自己所知道的。

写书是一项团队工作，我幸运地遇到了一个非常出色的团队，任何作者都会期盼与这种高水平的团队合作。感谢技术审稿人 Bill von Hagen 和编辑 Michelle Lowman 与 Frank Pohlmann。所有编辑都认真阅读了我写的内容并提供了反馈，他们的工作大大提高了本书的质量。我还要感谢 Apress 的产品团队，他们制作了封面，对各个页面进行了排版，并不可思议地把一堆文档变成了读者手中的这本书。这里要特别感谢 James Markham，他容忍了我无法在最后期限提交作品的行为，其耐心令人钦佩。最后，开源社区教会了我有关 Linux 的知识，邮件、新闻组和源代码本身都对我很有帮助。

前　言

在大约十年前进入嵌入式 Linux 领域时，我面对的一个问题是：我是不是应该使用操作系统？选择 Linux 意味着经常需要将该操作系统移植到目标硬件上运行，并联编移植所需的工具。随着时间的流逝，事情有了很大的变化，对许多项目来说 Linux 已经是默认选项，而需要决策的是在具体项目上应该使用该操作系统的哪些特性。如今的问题是：我应该如何配置我的 Linux 发布版？用技术术语来说，这是开发者态度在很短的时间范围内所发生的巨大转变。

在嵌入式领域，Linux 是如此普遍，以至于嵌入式处理器和开发板默认情况下都附带了 Linux。买家都期望开发板会启动 Linux，并带有随硬件一同提供的、进行嵌入式开发所需的工具。不同于 Linux 发展的早期阶段，现在的开发者无需把 Linux 移植到相应的开发板上，只需要对一个已经能够运行的 Linux 内核和根文件系统进行配置，使之适合应用的需求即可。

着眼于这样的背景，我以一个用户的视角来撰写本书：他即将开始其项目，而手头已经有一个可以在开发板上运行的 Linux 发布版。这不见得是最终随产品交付的 Linux 发布版，但很可能作为开发的起点。本书描述了诸如从头开始构建交叉编译器之类的任务，以便读者理解，但读者或许会使用与开发板一同提供的交叉编译器，这样读者就可以集中精力开发实际的应用。但学习构建和配置用于 Linux 系统的工具绝非浪费，当读者需要从你的系统中压榨出内存的每一位时，这些就成为基本功了。

更进一步，对于新的片上系统（System on a Chip, SOC）式设计，开发板附带的 Linux 发布版带有芯片上的设备所需的所有驱动程序。需要开发驱动程序的情形是比较罕见的。这意味着，大多数工程师把时间花费在内核的定制上，而不需要构建新的内核组件，而花费在所有内核配置和开发方面的总时间，只是过去几年的几分之一。

随着嵌入式设备中的处理器变得更为强大，开发者逐渐发现可以使用 C 以外的语言进行开发。将 C++用于开发已经很常见，还有其他高级语言如 Python、TCL、Java，甚至 Perl 或 PHP。对于使用汇编语言开始嵌入式开发的人来说，在嵌入式目标机上使用类似 Perl 的语言几乎是异端（如果不算是叛教的话）。但这些高级语言可以大大提高嵌入式开发的效率。在上市时间至关重要的环境下，高级语言显然会成为主流。

嵌入式项目的开发过程不同于其他的软件项目。首先会有一个设计过程，这涉及建立一个专用的容器，以及一个用户界面（可能是一个小的 LCD 显示器和一些按钮）。但更重要的是软件的部署和更新。项目的代码不会放到 CD 或网站上，而必须下载到带有 Linux 发布版的目标板上。在软件最初安装后，可能需要将其更新为包含 bug 修正或其他特性的新版本。根据系统的配置方

式，更新系统本身可能会成为另一个项目。

如果读者是在工作中开始一个嵌入式项目，或购买了某种面向计算机业余爱好者的比较廉价的开发板，我预祝你的努力能够获得好运。与以前任何时候相比，目前在嵌入式项目中使用 Linux 都更为容易，且有更多乐趣。

目　录

第1章

嵌入式 Linux 简介

Linux 是一种不可思议的软件。它是一种操作系统，当运行在 IBM 的 z 系列超级计算机上时，与运行在手机、工业制造设备、网络交换机，甚至挤奶机上相比，几乎没什么区别。更不可思议的是，该软件当前由数以千计最优秀的软件工程师维护，可以免费获得。

Linux 最初不是一个嵌入式操作系统。Linux 由一位芬兰的大学生（Linus Torvalds）创建，此人足够聪明：使其工作能够为所有人使用，从他人处获得输入，而最重要的一点是，将工作委托给其他有才能的工程师。随着该项目的增长，它吸引了其他有才能、能够对 Linux 作出贡献的工程师，这提高了这个急速增长的项目的价值和知名度，形成了一个自我支撑的正向循环，一直延续至今。

Linux 最初是为 Intel IA-32 体系结构编写，首先移植到摩托罗拉处理器。移植过程非常困难，这使得 Linus Torvalds 决定重新考虑内核的体系结构，使之能够比较容易地移植，最终在软件的处理器相关部分和体系结构无关部分之间建立了一个干净的接口。这种设计决策为 Linux 移植到其他处理器铺平了道路。

Linux 只是一个内核，它本身不会很有用。嵌入式 Linux 系统，或任何其他 Linux 系统，都需要使用许多来自其他项目的软件，以形成一个完整的操作系统。Linux 内核主要使用了 C 语言（和一些汇编语言），以及 GNU 工具集进行编写，如 make、GCC 编译器、为内核提供接口的程序，以及很多其他程序，读者在本书中会看到。在 Linux 创始的时候，这些软件大部分都已经存在，而幸运的是，其中大部分在编写时都考虑到了可移植性。这些软件或者可以直接用于嵌入式系统，或者在修改后适合于部署到嵌入式环境下，这一点对 Linux 在桌面计算机以外的设备上的普及度，贡献颇多。

说明 Linux 能够存在，很大程度上是因为 GNU（Gnu's Not Unix）项目，该项目意在开发一个开源的 Unix 实现。GNU 项目提供了一个高质量的编译器，还有命令行 make 环境，以及预期在类 Unix 系统上会有的那些基本实用程序。

本书将使读者逐渐熟悉嵌入式项目中对 Linux 的使用。因为 Linux 和相关的项目是开源的，读者需要学习如何从头开始构建一个嵌入式项目所需的一切东西。整个 Linux 环境已经发展到了这样一种程度：这些从头开始的构建活动不再是一种堂吉诃德式的训练，对任何愿意付出合理时间和工作量的工程师来说，这些构建工作都不成问题。对于建立一个小型的 Linux 系统来说，构建一个完整的 Linux 系统是最佳的训练，因为这样做的效果绝对会强于那种只是鼓舞士气的训练。

1.1　为何使用嵌入式Linux

嵌入式Linux与在全世界的数百万桌面计算机和服务器上运行的Linux发布版颇为相似，但它适用于特定的使用场合。在桌面计算机和服务器上，内存、处理器周期、用电量和存储空间都是有限资源，但在嵌入式设备上这些资源的受限程度要极端得多。在配置桌面计算机或服务器时，若干MB或GB的额外存储空间可能只是舍入误差。在嵌入式领域，资源是很重要的，因为它们决定了设备的单位成本，而该设备生产的数量可能以百万计。另外，额外的内存可能需要额外的电池供应电力，这又增加了重量。时钟速度较高的处理器会产生更多的热量。一些环境对发热量的限制很严格，因此所采用的散热设备非常受限。因而，在嵌入式程序设计中（无论是使用Linux还是其他的操作系统），大部分工作都集中在利用有限资源完成尽可能多的事情上。

与其他嵌入式操作系统相比，如VxWorks、Integrity和Symbian，Linux并不是最“苗条”的选项。一些嵌入式应用使用类似ThreadX[①]的框架，该框架直接运行在硬件上，完全避开了操作系统。另外，还可以绕开框架，编写直接运行在处理器上的代码。使用传统的嵌入式操作系统和Linux，最大的区别在于内核和应用程序的分离。在Linux下，应用程序所运行的上下文，是与内核完全分离的。除了内核分配的内存和资源以外，应用程序无法访问其他内存或资源。这种级别的进程保护意味着，有缺陷的程序会与内核和其他程序隔离开来，从而建立一个更安全且不易破坏的系统。这种保护是有代价的。

与其他方式相比，这种保护会增加资源开销，而采用Linux则进一步增加了这种开销。这意味着，从事嵌入式项目的工程师需要考虑，Linux增加的开销是否值得。近年来，SOC（system-on-chip，片上系统）处理器的成本和电力需求量已经降低到可以与过去的低功率8位微控制器相比，因而使用更复杂的处理器已经是一个可行的选项。许多设计方案使用现成的SOC处理器，无需以太网芯片、显示芯片或其他不使用的组件。

Linux之所以能够繁荣起来，是因为它提供了其他嵌入式解决方案所缺乏的能力和特性。这些能力对于实现更为复杂的设计非常必要，这些设计能够区分当前市场上的各种设备。Linux的开源性质，使得嵌入式工程师可以利用开源社区不断产生的开发成果，任何单独的软件厂商都无法跟上开源社区的脚步。

1.1.1　使用嵌入式Linux的技术原因

Linux在技术上的质量，推动了对它的采用。Linux不只是Linux内核项目。相关的软件也处于技术发展的最前沿，这意味着无论是解决当今的技术问题，还是在可预见的未来，Linux都是正确的选择。

例如，嵌入式Linux系统包括以下软件。

- SSL/SSH：OpenSSH项目是现在最常用的加密和安全机制。该项目的开放性，意味着有数以千计的安全专家在不断地评估它。在发现问题时，如果攻击本身没有包含更新，更新会

① 我不能评论ThreadX的实用性或特性，但我可以确定，过去10年间每次商业展览期间他们分发的填充毛绒玩具猴十分可爱。该公司还赞助了一个摊位一年的时间，由穿着《星际迷航》（Star Trek）服装的演员来解释ThreadX能做什么。我没怎么注意，但在演出后拿到了另一个玩具猴。

在数小时内出现。

- Apache 及其他 Web 服务器：Apache Web 服务器适用于需要全功能 Web 服务器的嵌入式设备。对于没那么高要求的设备，用户可以选择更小型的 Web 服务器，如 Boa、lighttp 和（笔者个人爱好）micro_httpd。
- C 库：Linux 环境在此领域有大量的选项可用，从全功能的 GNU C 库到最低要求的 dietlibc。如果读者对嵌入式 Linux 开发不那么熟悉，进行此项选择可以强化开源代码的开放性。
- Berkeley 套接字（IP）：许多项目从其他操作系统迁移到 Linux 是因为 Linux 包含完整的、高性能的网络协议栈。网络化的设备已经成为常规，很少有例外。

以下各节将解释为什么 Linux 操作系统在技术上最适合于嵌入式开发。

1. 基于标准

Linux 操作系统和相关的开源项目遵守工业标准。大多数情况下，可用的开源实现是某个标准的规范或参考实现。参考实现包含了对规范的解释，是一致性测试的基础。简言之，参考实现是用于度量其他实现的标准。

如果读者不熟悉参考实现的概念，可能会有点困惑。例如，POSIX（Portable Operating System Interface for Unix）中用于处理线程和进程间通信的部分，通常称作 pthreads。POSIX 标准组是 IEEE（Institute of Electrical and Electronics Engineers，电气和电子工程师协会）的一部分，它是一个委员会，设计了用于与线程交互的 API，但该标准的实现则由另一个标准组完成。实际上，在标准的工作开始时，委员会的一个或多个参与者会自愿创建相关的代码，即参考实现。参考实现会包括一个测试套件，其他实现需要通过测试套件，才能证明相关代码是按照规范编写的。

使用基于标准的软件，不仅关系到质量，也关系到平台无关性。项目基于遵守标准的软件，可以减少因为特定于厂商的特性而锁定到具体平台的可能性。厂商可能是善意的，但这些额外特性的好处通常会因为缺乏互操作性和自由选择权而被抵消，互操作性和自由选择权已经悄然成为商业事务的一部分，但很少被认真考虑。

在一个有许多嵌入式设备建立连接（很多时候是连接到任意的系统，而非彼此连接）的世界里，标准变得越发重要。以太网是一种这样的连接方法，但还有其他选择，如 Zigbee、CANbus、SCSI，这只是其中几个例子而已。

2. 进程隔离与控制

Linux 内核在最基本的层面上，以通用 API 的形式提供了这些服务，供访问系统资源之用。

- 管理任务，使之与内核彼此隔离。
- 提供统一接口，供访问系统硬件资源之用。
- 当存在争用时，充当资源的仲裁者。

这些是非常重要的特性，与硬件和资源访问未能得到严密管理的环境相比，这些特性将产生一个更为稳定的环境。例如，在没有操作系统时，每个运行的程序对所有可用的物理内存都具有同等访问权限。这意味着，一个程序中的溢出错误，将会写入到由另一个程序使用的内存中，除非对系统中所有的代码进行检查，否则第二个程序的失败原因会看似神秘而无法解释。资源争用的概念是比较复杂的，不仅仅只是确保两个进程不会同时向串口写数据这样简单，稀缺的资源是处理器时间，而操作系统可以决定在何时运行哪个进程，以便最大化所能处理的工作量。以下各

节将更详细地介绍上述各项。

● 管理和隔离进程

Linux是一个多任务操作系统。在Linux中，进程这个词描述了一个任务，内核将跟踪该任务的执行。多任务意味着，内核必须保存一些数据，记录哪个任务处于运行状态、任务的当前状态、任务使用的资源，如打开的文件和内存。

对于每个进程，Linux都会在进程表中创建一个对应项，并为该进程分配一个独立的内存空间、若干文件描述符、寄存器值集合、栈空间，以及其他与进程相关的信息。在进程创建之后，进程不能访问另一个进程的内存空间，除非二者建立了一块共享内存区。但对共享内存区的访问并不意味着可以访问另一个进程中的任意地址。

Linux中的进程可以包含多个执行线程。一个线程可以共享创建该线程的进程的内存空间和资源，但它具有自身的指令指针。线程不同于进程，可以访问彼此的内存空间。对于某些应用来说，这种资源共享是需要的，而且比较方便，但管理几个线程对资源的争用，本身就是一项研究。使用Linux，重要的一点是，你具有设计上的自由度，可以选择如何使用这些进程控制结构（即线程）。

进程不仅彼此隔离，而且也与内核隔离开来。进程同样不能访问内核中的内存。进程对内核功能的访问是发生在可控制环境下，例如syscall或文件句柄。syscall是system call的缩写，即系统调用，是操作系统设计中的一个通用概念，它允许程序发出一个调用，执行内核中的某些代码。对于Linux来说，为方便起见，用于执行系统调用的函数就命名为`syscall ()`。

在读者使用syscall时，本章后面会介绍，其工作方式与普通的API函数调用非常相似。使用文件句柄，可以打开文件来读写数据。文件的实现归结为一系列syscall，但文件语义使得这些系统调用在某些情况下很容易使用。

进程与内核的完全隔离，意味着无需再调试与进程彼此覆写内存导致的问题或访问共享资源（如串口或网络设备）导致的竞态条件。另外，操作系统的内部数据结构与用户程序的隔离，使得不会再出现应用程序错误导致整个系统停机的问题。Linux本身所提供的这种抗破坏性，是一些工程师选择Linux，放弃更轻量级解决方案的原因。

● 内存管理和Linux

Linux使用一种虚拟内存管理系统。在20世纪60年代早期，虚拟内存的概念就已经出现，概念本身很简单：进程将其内存视为一个字节向量，在程序读写内存时，处理器连同操作系统会将程序使用的虚拟地址转换为物理地址。

处理器中执行该转换的部件称为MMU（memory management unit，内存管理单元）。在进程请求内存时，CPU查找由内核填充的一个表，将所请求的虚拟地址转换为物理地址。如果CPU无法转换该地址，则引发一个中断，将控制传递到操作系统以解析该地址。

由内存管理单元提供的间接性意味着，如果进程请求的内存超出其界限，操作系统会得到通知并进行处理，或将该情况传递给发生问题的进程。在缺乏适当内存管理的环境中，进程可以读写任意物理地址。这意味着，内存访问错误可能会被悄然略过，直至程序的某个部分因为内存被另一个进程破坏而失败。

运行在Linux中的程序访问的是虚拟内存空间，即在程序运行时，其地址空间是整个系统内存的一个子集。该子集看起来起始于0。实际上，操作系统会为相关进程分配一份内存并配置处

理器，使得运行的程序认为地址 0 是虚拟地址空间的起点，但虚拟地址空间可以起始于物理内存中的任意位置。对使用分页机制的嵌入式系统来说，这一假象仍然会继续存在：内核将某些不使用的物理内存的内容换出到磁盘，该特性通常称为*虚拟内存*。许多嵌入式系统不使用虚拟内存，因为系统中不存在磁盘。但对于有磁盘的系统来说，该特性将 Linux 与其他嵌入式操作系统区分开来。

- *访问资源的统一接口*

因为有许多种资源存在，这一点听起来有点不靠谱。考虑最常见的资源：系统内存。在所有的 Linux 系统中，从应用程序的视角来看，堆内存都是使用 `malloc()` 函数分配的。例如，下述代码将分配 100 字节内存，并将第一个字节的地址存储在 `from_the_heap` 中：

```
char* from_the_heap;
from_the_heap = (char*) malloc(100);
```

无论系统底层由何种处理器来运行代码，或处理器如何访问内存，上述代码都可以在所有 Linux 系统上工作（或以可预测的方式失败）。如果启用分页虚拟内存（即某些内存可能存储在一个物理设备上，如硬盘），在进程请求某个地址时，操作系统会确保相应的地址存在于物理内存中。

内存管理需要操作系统和处理器之间的相互作用，才能正常运作。Linux 的设计，使得在所有支持的处理器上都可以用同样的方式访问内存。

访问文件同样如此：只需打开一个文件描述符，然后即可读写数据。内核会处理字节数据的取出或写入，无论物理设备如何处理该数据，程序层面上的操作都是同样的：

```
FILE* file_handle;
file_handle = fopen("/proc/cpuinfo", "r");
```

因为 Linux 基于 Unix 操作系统的哲学“万物皆文件”，所以访问系统资源最常用的接口就是通过文件句柄。无论相关功能在底层硬件上如何实现，操作文件句柄的接口都是相同的，甚至 TCP 连接也可以用文件语义表示。

资源访问的统一，使得我们可以在开发系统上仿真目标机的环境，这个过程曾经需要专门（有时候比较昂贵）的软件。例如，如果目标设备使用了 USB 子系统，那么访问该子系统的接口在目标机和开发机上是相同的。如果读者所处理的设备需要跨越 USB 总线传输数据，那么代码可以在开发用的宿主机上开发、调试、测试，与调试远程目标机上的代码相比，这个过程要更为容易和快速。

- *系统调用*

除了文件语义，内核也使用系统调用的思想来向外公开一些功能。系统调用是一个简单的概念：当使用内核并打算向外公开一些功能时，只需要在一个向量中增加一项，指向相关例程的入口点。进行系统调用时，来自应用程序内存空间的数据会复制到内核的内存空间。所有进程的所有系统调用都通过同样的接口进行。

在内核完成系统调用时，它会将结果传回调用者，将结果返回到应用程序的内存空间。使用这一接口，用户空间中的程序无法访问内核中的数据结构。内核也可以对其数据保持严格的控制，避免错误调用导致的数据损坏。

3. 外设支持

根据最新的统计，Linux支持200种以上的网络适配器、5家厂商的闪存和10种USB海量存储设备。因为SOC厂商使用Linux作为其芯片的测试系统，对芯片本身的支持就隐含了对设备上各组件的支持。

广泛的设备支持只是下述事实的一个结果：Linux运行在数百万桌面计算机和服务器上，这代表的用户群体是不可能被设备厂商忽视的。另外，Linux的开放性使得设备厂商不用从操作系统厂商获取开发许可即可创建驱动程序。

真正把Linux对外设的支持与其他嵌入式操作系统区分开的是，驱动程序（主要）都是以C语言编写，使用内核提供的API来实现其功能。这意味着，一旦为x86版的Linux内核写好了某个驱动程序，不需要什么工作量（或只需极少的工作量）就可以将其迁移到不同的处理器。

4. 安全

安全意味着控制对机器上数据和资源的访问，以及对计算机所处理的数据保密。Linux安全性的关键在于其开放性。任何（每个）人都可以审阅源代码，因而，所有人都可以看到、理解、并修复安全漏洞。

安全性有几个不同的方面，对嵌入式或任何其他系统来说，所有这些可能都是必要的。首先是确保用户和程序对资源具有最低水准的权利，以利于执行；其次是一直隐藏信息，直至具有正确凭据的用户请求查看或修改该信息。Linux的优点在于，所有这些工具都是免费的，因此读者可以选择满足项目需求的正确工具。

- SELinux

若干年前，一个对安全感兴趣的政府部门NSA（National Security Agency，美国国家安全局）和几个有着类似兴趣的私营公司径自决定检查Linux内核并根据一个MAC（Mandatory Access Control，强制访问控制）模型向其中引入数据保护、程序隔离和安全策略等概念。该项目称作SELinux（SE表示Security Enhanced，即安全增强），该项目所做的改动和引入的概念成为了Linux内核2.6.0版的一部分。

SELinux中的MAC概念指定，程序必须被指派某些权限才能进行相应的活动，如打开套接字或文件，这是其安全策略的一部分。这种指派必须由管理员进行。系统的普通用户不能进行修改。SELinux系统按最小特权原则（principle of least privilege）运作，这意味着进程只会被授予必需的权限，绝不会过多。最小特权概念的作用在于，它使得错误或危及安全的程序能够在正确配置的环境下变得不那么危险，因为管理员已经授予了程序正常运作所需的最小权限集。读者可能会猜到，创建安全策略本身就是一个项目。我会花费些时间，来讨论如何在嵌入式系统上创建安全策略。

- PAM

PAM（Pluggable Authentication Module，可插拔认证模块）为认证用户的过程建立了一个统一的接口。在Linux系统上进行用户认证时，需要在`/etc/passwd`文件中查找用户名并检查其中保存的加密口令（或使用shadow口令文件）。PAM框架还提供了会话管理功能：在用户认证之后，系统注销之前，执行某些操作。

对于目标设备需要连接到公司网络环境的嵌入式项目来说，PAM系统的开放式设计非常重