Lenix
嵌入式操作系统

罗 斌 编著

北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

Lenix 嵌入式操作系统

罗 斌 编著

北京航空航天大学出版社

内 容 简 介

本书分4个部分介绍Lenix，首先用具体的例子向读者展示部分Lenix的能力，然后对Lenix涉及的操作系统的基本概念进行说明，接着再对Lenix的整体结构和引入的CPU、计算机模型进行介绍，最后用大量篇幅对进程管理、时间管理、内存管理、IPC、设备管理、人机交互和移植等几个部分的设计思路、API和源代码进行详细的分析。

本书适合普通高校计算机专业本科生及想了解操作系统工作原理的读者，以及希望掌握一个嵌入式系统或者学习如何开发操作系统的读者。

图书在版编目(CIP)数据

Lenix嵌入式操作系统 / 罗斌编著. -- 北京 : 北京航空航天大学出版社，2014.7

ISBN 978-7-5124-1421-1

Ⅰ. ①L… Ⅱ. ①罗… Ⅲ. ①实时操作系统 Ⅳ. ①TP316.2

中国版本图书馆CIP数据核字(2014)第101994号

Lenix嵌入式操作系统

罗 斌 编著

责任编辑 宋淑娟

*

北京航空航天大学出版社出版发行

北京市海淀区学院路37号(邮编100191) http://www.buaapress.com.cn

发行部电话：(010)82317024 传真：(010)82328026

读者信箱：emsbook@gmail.com 邮购电话：(010)82316524

涿州市新华印刷有限公司印装 各地书店经销

*

开本：710×1 000 1/16 印张：22.75 字数：485千字

2014年7月第1版 2014年7月第1次印刷 印数：3 000册

ISBN 978-7-5124-1421-1 定价：59.00元

前 言

随着计算机技术的发展，计算机在日常生活中的应用越来越广泛。从最早在科学计算领域的应用，发展到个人电脑普遍进入家庭，再到冰箱、电饭煲等小电器也嵌入了计算机系统，至此人们似乎对计算机系统已经不再陌生，但细想起来，绝大多数人仅是知道计算机由多个硬件组成，稍微了解多一些的人还会知道这些硬件发挥作用需要操作系统。

操作系统管理着计算机系统的软、硬件资源，可以说它是计算机系统的灵魂。这个灵魂从诞生的那天起就一直保持着神秘。因为它的神秘，使得很多人对其工作原理和方式产生了浓厚兴趣，并想了解其中的奥秘。但现有的操作系统，要么无法获得源代码，例如 Windows；要么在经过多年的发展后，已变得庞大而复杂，比如 UNIX、Linux，再加上代码上的优化和各种初始准备条件，更让人难以摸到其核心。因此，对于操作系统爱好者来说，如果有一个相对简单，但功能还算齐全，又可以获得源代码的操作系统，就是一大幸事。对于母语是中文的爱好者来说，有一份中文注释的操作系统源代码则更好。

Lenix 就是在这样的想法下开发的。为了简化开发，Lenix 选择常见的 PC 作为开发平台，而且是在 16 位实模式下开发的。对于绝大多数操作系统爱好者来说，如何学习操作系统是个很大的难题，特别是开发 PC 操作系统，绝大部分人会认为要进入保护模式才能算是操作系统。很不幸，这是一个极大的误区。进入保护模式是为了发挥 CPU 的功能，而不是操作系统的核心，操作系统的核心是管理计算机系统的软、硬件资源。保护模式只是操作系统发挥 X86 系列 CPU 功能的一个方式。从学习的角度来看，保护模式反而干扰了对操作系统的进程管理和内存管理这些核心内容的学习。因此，实际上在 16 位实模式条件下才能更直观地接触到操作系统的关键内容。

开发 Lenix 的过程比较顺利。因为在此之前，笔者已经开发了一个 32 位的通用操作系统，名为 SHEMOX。SHEMOX 已经实现了自启动、进程管理、虚拟内存管理、文件系统，还支持 PE 格式的可执行文件和动态链接库。Lenix 从 2011 年中开始开发，到系统跑起来，大概只用了 2 个星期。但后续功能的扩展和调试，由于出差、外出学习等事情，断断续续花费了一年多的时间。

在开发 Lenix 的过程中，代码编辑器使用的是 Visual Studio 2012 的 express 版本，在 Visual PC 2007 上进行调试，没有使用物理机器作为调试平台。当然，最关键的还是要找到一个 16 位的 C 语言编译器，笔者使用的是 Borland C/C++ 3.1。既然选择了 BC31，那么自然要使用 16 位的操作系统作为辅助，因此自然选择了 DOS，实际上这是 Windows 98 制作的启动系统。

Lenix 这个名称本来的意义是“学习用的操作系统”，因为这个系统就是为了便于学习而开发的。命名时就用了英文单词“Learn”作为名称，完全符合本意。随后看到很多操作系统名字的末尾都有一个“x”，给人一种探索未知领域的感觉，很科幻，因此就凑趣，保留 Learn 的头尾，在末尾加了 x，但感觉“Lenx”这个词没有美感，就又加了一个“i”，也就是笔者自己，于是就有了 Lenix 这个名称。随着开发工作的深入，功能逐步增加，这时笔者发现其在嵌入式领域中已经具备了一定的可用性，因此就对名称的意义进行了重新阐释：名称不变，还是 Lenix，但其代表的意义变更为“罗氏嵌入式操作系统(Luobin's Embedded OS)”，也就是用笔者的名字来命名这个系统，且将其定位在嵌入式领域。

本书面向有一定 C 语言、汇编语言编程经验和 PC 硬件知识的读者。如果没有以上基础，但又确实对操作系统感兴趣，那么在阅读本书前，最好先学习一定的 C 语言和汇编语言编程知识，并初步了解 PC 的硬件配置。这样会更易理解书中的内容。

关键章节内容采用的结构是先说明需求和设计等基本情况，然后安排 API 的用法说明和一些程序例子，最后对功能的实现做详细解析，包括数据类型、公共变量和源代码的说明。笔者认为这样的方式比较便于理解和学习。

各章节的主要内容如下：

第 1 章，通过具体的演示程序来展示 Lenix 的功能，使读者对 Lenix 建立一个比较直观的认识。

第 2 章，介绍操作系统的基础概念，这些概念是在 Lenix 中采用的。对这些概念建立起统一的认识，将有助于理解 Lenix。

第 3 章，从整体上介绍 Lenix，使读者建立起较为系统的认识。

第 4 章，说明 Lenix 的临界段保护方法，这是开发操作系统的关键基础，会一直伴随开发的整个过程。

第 5 章，说明 Lenix 引入的硬件模型。模型包含 CPU 模型和计算机模型，各模型都定义了一定数量的接口，这些接口为实现操作系统的功能提供了便利。

第 6 章，说明 Lenix 进程管理的设计和实现。对于单个进程，进程管理的主要工作是对进程生命周期的管理；对于多个进程，主要工作则是关注进程如何被调度，即如何分配 CPU。

第 7 章，说明 Lenix 时间管理的设计和实现。时间管理通过时钟中断来提供一个基本的计时依据，并在此基础上开发了定时器等功能。

第 8 章，说明 Lenix 内存管理的设计和实现。系统的内存总是无法满足程序的

需要，因此提供了动态内存管理功能。系统还提供了高效的定长内存管理和适用广泛的堆内存管理。

第 9 章，说明 Lenix 的 IPC 设计和实现。系统实现了自旋锁、普通锁和互斥对象三个基本的 IPC 机制，还提供了邮箱来完成进程间少量数据的通信。

第 10 章，说明 Lenix 设备管理的设计和实现。系统定义 Lenix 驱动模型（LDM），包含设备驱动接口（DDO）、设备管理的框架、设备使用规范和驱动程序框架。

第 11 章，说明 Lenix 人机交互的设计和实现。人机交互是使用计算机系统的重要组成部分，目前系统提供了利用 TTY 终端与 SHELL 解释程序组合的人机交互方式。

第 12 章，说明 Lenix 的移植。通过在 16 位 PC 上的开发来说明如何移植 Lenix。

在本书编写过程中，遇到的问题除通过自己试验摸索解决外，其他都是从网络上获得了答案或者灵感，这里感谢在网络上共享知识的人们。

笔者已尽最大努力来减少书中的错误，但仍在所难免。如果您发现了错误，请告诉笔者，笔者会及时发布更正通告，并在本书下一版中进行改正。如果您对本书的内容有疑问，欢迎发电子邮件给笔者，笔者将尽力给予回复。邮箱：sourcex. robin@gmail. com。也可以访问笔者的网站，以了解更多信息。网址：http://www. soruce-ex. com。

罗 斌

2014 年 5 月

目　录

第1章 引例

在讨论 Lenix 前，先用几个演示程序来展示 Lenix 的一些基本功能。读者可以通过这些演示程序对 Lenix 的功能及编程框架建立起一个初步的认识。

这些演示程序使用 BC31 编译，运行于 PC 平台上。虽然演示程序可以在 Windows XP 及以前的操作系统中运行，但还是强烈建议读者在 PC 模拟器中使用 DOS 来运行演示程序。PC 模拟器有很多种，读者可以选择自己习惯的模拟器。本书采用的是微软公司的 Virtual PC 2007，原因是它免费及笔者的习惯。

1.1 多进程演示

多进程是 Lenix 的基本特征，因此选择了多进程能力作为第一个演示内容。该演示向读者展示了 Lenix 的多进程和分配 CPU 运行时间的能力。

读者可以在 demo\pc\proc1 目录下获得该演示程序的源代码。在 DOS 下使用同一目录下的 Makefile 编译和链接，然后运行得到的程序就可以看到运行结果。编译的方法可以参考附录 A。

1.1.1 演示内容

这里通过输出一个表格来展示相应的能力，表格的内容是进程的一些信息，包括：

- pid：进程编号。
- pr：进程优先级。
- pn：进程优先数。
- run time：进程运行时间，以时钟节拍为单位。
- use：进程运行时间占系统总运行时间的百分比。
- test：一个不断变化的信息，表示进程在运行。

演示一共建立 7 个无限循环的进程，分别命名为 app1～app7。这些进程分别向屏幕的固定位置输出一行字符串，这些输出的信息组合起来就形成了一个表格。其中，app1 的作用是显示表头，app2～app7 的作用是输出表格的内容，每个进程输出一行。

为了更清晰地看到系统运行的状态，演示程序额外输出了一些调试信息。系统会在每次时钟中断时，在屏幕右上角第一行输出系统运行的累计时钟节拍数，同时在

屏幕的右下角，由下往上，用绿色字符输出当前系统中所有进程的信息，输出的信息包括进程编号、优先级、栈指针、调度因子和进程名。当发生键盘中断时，会在屏幕右上角第二行输出键盘的状态、按键对应的字符、ASCII 码和扫描码。在每次发生调度时，在屏幕右下角最后一行用品红色(红蓝混合)输出累计的调度次数。当然，本书中的所有演示都会输出这些调试信息。

演示程序运行结果如图 1.1 所示。

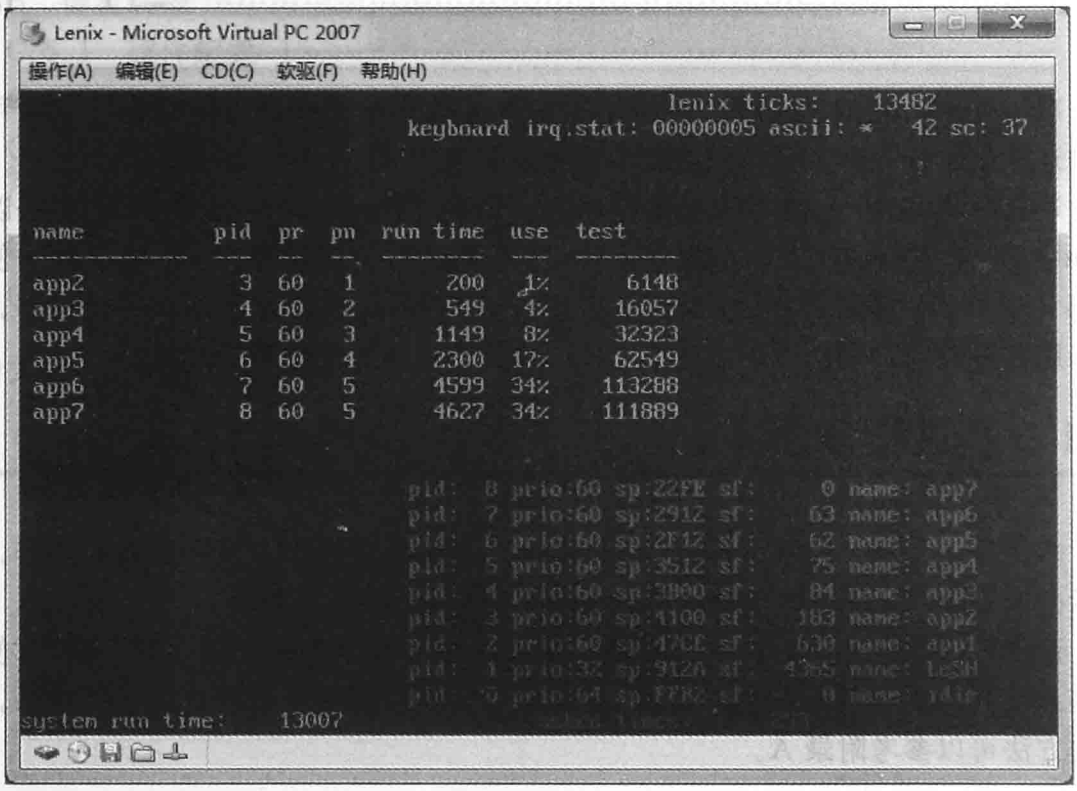

图 1.1 多进程演示运行结果

从图 1.1 可以看出，app6 和 app7 的运行时间最长，其他进程依次递减，app2 进程的运行时间最短。这几个进程都处于运行状态，只是由于进程优先数的不同，它们获得的运行时间也不同。从这里即可以看出 Lenix 的多进程能力和 CPU 时间的分配能力。

1.1.2 程序说明

按照演示的内容，输出的表格可以分为输出表头和输出表格内容两个部分。因此要分别为这两个部分引入 Title 和 Content 函数。

Title 函数输出表头，方式是每隔 1 000 ms 输出一次，也就是每秒刷新一次表头。进程通过在输出表头后使进程延迟的方法来实现这一要求。

Content 函数输出表格内容。为了表示进程正在运行，设置了一个计数器，进程在每次循环中都会递增计数值，并在表格最后一列输出其值。由于 Content 函数一

直处于运行状态，因此表格的输出内容也就处于不断的变化之中，从而达到显示进程运行结果的目的。

程序中的 main 函数完成系统的初始化，User_initial 函数完成用户程序的初始化。这些是 Lenix 应用程序编程框架要求提供的函数。演示程序的核心部分是Title 和 Content 函数。

多进程演示程序的源代码见程序 1.1。

程序 1.1

-- demo\pc\proc1\demo.c --

```
 1 #include<lenix.h>
 2
 3 #define USER_APP_STACK 2048
 4
 5 byte_t app_stack1[USER_APP_STACK];
 6 byte_t app_stack2[USER_APP_STACK];
 7 byte_t app_stack3[USER_APP_STACK];
 8 byte_t app_stack4[USER_APP_STACK];
 9 byte_t app_stack5[USER_APP_STACK];
10 byte_t app_stack6[USER_APP_STACK];
11 byte_t app_stack7[USER_APP_STACK];
12
13 void Clk_msg(void);
14
15 void Title(void * param)
16 {
17     char msg[80] = {0};
18
19     param = param; /*  避免编译器发出变量未使用警告  */
20
21     for(;;)
22     {
23         Con_write_string(0,5," name          pid  pr pn  run time  use  test",
0x7);
24         Con_write_string(0,6," ------------ --- -- -- ---------- --- --------
-",0x7);
25         _sprintf(msg,"system run time: %8ld",Clk_get_ticks());
26         Con_write_string(0,24,msg,0x7);
27         Proc_delay(1000);
28     }
29 }
30
```