Linux 系统中内核 logo 独立于内核文件的研究与实现

赵国伟[1]，唐小琦[1]，宋　宝[1]，朱　建[2]
（1. 华中科技大学机械科学与工程学院，湖北 武汉 430074；2. 苏州意立达智能科技有限公司，江苏 苏州 215144）

Research and Realization of Independence of Kernel Logo From Kernel Files Based on Linux

ZHAO Guo-wei[1]，TANG Xiao-qi[1]，SONG Bao[1]，ZHU Jian[2]
（1. School of Mechanical Science and Engineering，Huazhong University of Science and Technology，Wuhan 430074，China；
2. Suzhou Elite Intelligent Technology Co.，Ltd.，Suzhou 215144，China）

摘要：针对目前 Linux 内核 Logo 文件都是编译到内核里面，更换必须重新编译内核这一现象，分析了嵌入式 Linux 系统中，内核 logo 的显示原理以及通用更换方法，提出了内核 logo 独立于内核文件，存放在 NAND Flash 固态存储设备上，实现显示和快速更换的方案。对方案进行了设计实现和测试，并成功应用在 ARM 嵌入式 Linux 数控系统中。

关键词：Linux；内核 logo；独立；NAND Flash
中图分类号：TP316
文献标识码：A
文章编号：1001-2257(2012)12-0003-04

Abstract: At present，the files of Linux kernel logo are compiled into the kernel which must be re-compiled when logo is changed. In view of this phenomenon，paper analyzed display principle and general replacement method of kernel logo in embedded CNC system，proposed the new plan to display and rapidly replace，in which logo is independent of kernel file and stored in NAND flash device. After design realization and test，plan successfully applied to ARM embedded Linux CNC system.

Key words: Linux；kernel logo；independence；NAND flash

0　引言

Linux 操作系统的产品标识是一只企鹅（Linux 内核 logo），通过将 logo 文件编译进内核实现。在嵌入式 Linux 系统中，当根据需求将 logo 更换时，需要更换内核文件中的 logo 文件，重新编译生成新的内核镜像文件。一方面开发应用过程变得烦琐，另一方面内核文件因 logo 不同而变动，难以实现版本的统一。因此，提出一种新的内核 logo 显示和更换方案。

收稿日期：2012-08-20
基金项目：国家科技重大专项（2012ZX04001012）；国家自然科学基金资助项目（50905069）；姑苏创新创业领军人才专项（ZXL2012034）

1　方案设计

1.1　默认内核 logo 显示原理

内核镜像无法存储图片格式的文件，只能将图片转化为数组，然后将数组编译进内核。内核启动，加载完显示驱动后将数组中的数据传给 FrameBuffer，从而显示出来。Linux 内核目前支持 pbm，pgm，ppm 格式（默认使用 ppm 格式）的图片文件，存放在内核文件 driver/video/logo/目录下。内核编译时，首先调用内核文件 scripts/pnmtologo/pnmtologo.c，将 driver/video/logo/目录下的 ppm 图片生成图片数据数组，存放在同目录下的 C 函数中，编译进内核。若更换内核启动 logo，需替换相应目录下的 ppm 文件，重新编译内核。默认内核 logo 显示过程如图 1 所示。

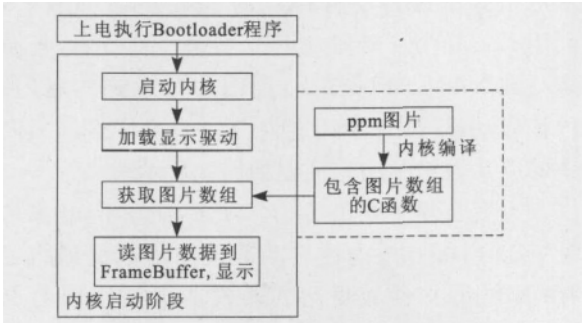

图 1　默认内核 logo 显示过程

1.2　内核 logo 独立于内核文件的设计

在嵌入式 Linux 系统中，各种文件存储在 Flash 等固态存储设备上。Bootloader 映像是系统上电执行的第 1 个程序，用来初始化硬件设备、准备软件环境，进而调用操作系统内核，内核启动后，会挂载根文件系统，进一步启动文件系统中的应用程序。Bootloader 阶段有启动加载和下载 2 种工作模式。加载模式下，Bootloader 自动从板子的固态存储设备上将操作系统加载到 RAM 中运行；下载模式下，系统自启动被中断，开发人员可以使用各种命令，通过串口或网络连接等通信手段下载更新文件，将它们直接放在内存运行或是烧入 Flash 等固态存储设备中[1]。

内核 logo 显示方案的基本思想：利用内核中的图片数组结构，将内核文件中静态的 ppm 文件删除，不再通过内核编译命令生成图片数组数据，而是内核启动过程中直接将预先烧写到 NAND Flash 中的 bmp 图片信息读到数组结构中。图片更新时，只需在 Bootloader 下载模式下更新 NAND Flash 中的 bmp 图片，logo 文件与内核文件分离，不再依赖于内核文件。具体方案如图 2 所示。

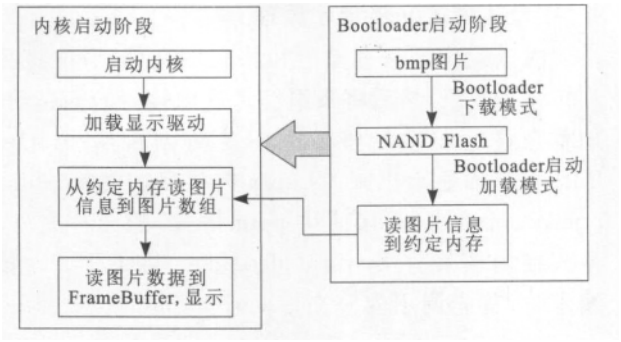

图 2　Logo 独立于内核方案设计

2　实现过程

在 PC 机上，通过 Photoshop 工具，将任意一张常用格式的图片文件保存为 224 色的 bmp 文件。利用 Bootloader 阶段 TFTP 网络传输技术，将图片烧写到 NAND Flash 中，读图片信息到约定内存。内核启动过程中，读约定内存中的图片信息到内核中的图片数组结构。流程如图 3 所示。

方案实现主要包括 3 个方面：烧写 bmp 文件到 NAND Flash 中；内核启动前读图片数据到约定内存；将 bmp 文件数据格式转换成内核通用的 logo 图片数组格式。嵌入式数控系统的 NAND Flash

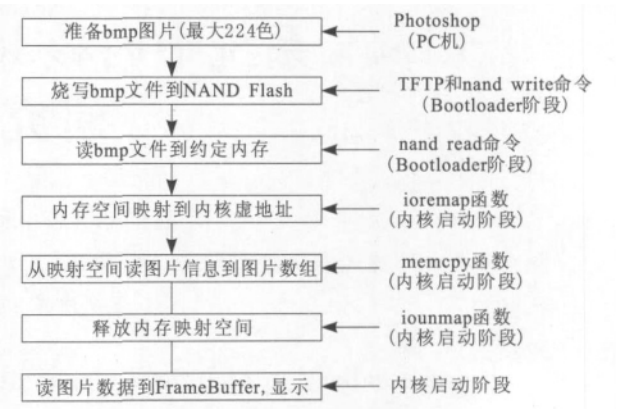

图 3　实现流程

大小为 64 MB，其中 0x160000 到 0x260000 的 1 MB 分区用来存放内核 logo 图片，内存地址 0x32200000 是 Bootloader 和内核约定的图片数据存放位置。

2.1　烧写 bmp 文件到 NAND Flash

系统引导程序 Bootloader 支持 TFTP 网络传输功能以及各种操作命令。在 Bootloader 下载模式下，用其支持的 TFTP 网络传输协议，烧写 logo 文件到 NAND Flash 分区。其中 run 命令可以执行环境变量中的复合命令；tftp [loadAddress] [bootfilename] 命令使用 TFTP 协议从 PC 机下载文件到内存；nand erase [off] [size]，从 NAND Flash 偏移地址 off 处擦除 size 个字节的长度；nand write [addr] [off] [size]，把开始地址 addr 内存中的 size 字节数据写到 NAND Flash 偏移地址 off 处[1]。设计中，Bootloader 阶段，首先建立新的环境变量 install-kernellogo，Bootloader 下载模式下，通过 run install-kernellogo 命令，将 bmp 文件放到 NAND 分区 160000 ~ 260000 中。

2.2　读图片数据到内存

内核 logo 显示是在内核加载完显示驱动后完成，先将图片数据加载到内存，再将内核镜像加载到不同内存处，之后启动内核。这一过程可以通过修改启动参数实现。

bootcmd 是系统自启动时执行的环境变量，可在里面设置启动命令参数：bootm [addr]，启动 addr 处的内核镜像；nand read [addr] [off] [size]，从 NAND Flash 偏移地址 off 处读 size 个字节数据，并存放到开始地址为 addr 的内存中。系统自启动通过调用环境变量 bootcmd，把 Flash 段 160000 ~ 260000 中图片数据放到内存 0x32200000 处，Flash 段 260000 ~ 460000 中的内核镜像放到内存

32000000 处，之后启动 0x32000000 处的内核程序。

2.3　读内存数据到图片数组

内核启动前，图片信息已经在约定的内存地址空间，在显示驱动加载完成后，通过 memcpy 函数将内存地址空间的图片信息读到通用图片数组中，再将数组中的数据传给 FrameBuffer，从而显示出来。其中，memcpy(void * dest, const void * src, size_t n) 用来拷贝 src 所指的内存内容前 n 个字节到 dest 所指的内存地址上。

3　关键技术

设计的关键技术是：系统启动时，显示驱动加载完成前，将约定内存中 bmp 图片信息转换成内核文件需要的图片数组，填充图片数组结构体。

3.1　内核 ppm 文件格式数组分析

内核默认的 logo 图片为 drivers/video/logo/目录下的 logo_linux_clut224. ppm，内核编译时，将其转化成 logo_linux_clut224. c 函数，其中包含 3 个数组文件，即

```
static unsigned char logo_linux_clut224_data [ ] __initdata = { };
static unsigned char logo_linux_clut224_clut [ ] __initdata = { };
const struct linux_logo logo_linux_clut224 __initconst = {
. type = LINUX_LOGO_CLUT224,
. width = xx,
. height = xx,
. clutsize = xx,
. clut = logo_linux_clut224_clut,
. data = logo_linux_clut224_data
}
```

logo_linux_clut224_clut 是调色板数组，每 3 个成员(RGB) 代表 1 种颜色。图片使用的颜色总数不能大于 224(内核源码决定) 。

logo_linux_clut224_data 数组的成员是指向调色板数组的索引，但是它的值比实际值大 32。比如第 1 个成员的内容为 0x20(用十进制表示是 32) ，所以它代表图片第 1 个像素的颜色，是调色板数组中 0，1，2 这个 3 个成员代表的一种颜色。索引值 index 与 RGB 的换算关系如下(clut 表示调色板数组) :

$$R = clut\ [(index - 32) \times 3]$$
$$G = clut\ [(index - 32) \times 3 + 1]$$
$$B = clut\ [(index - 32) \times 3 + 2]$$

由于索引值由 1 个字节表示，因此最多表示 256 种颜色。实际值又统一加了 32(0x20) ，所以只能表示 224 种颜色。

3.2　bmp 文件转化为内核图片数组

通过对内核 logo 图片数组分析，可知需要获得的图片信息包括调色板、调色板索引值(位图数据) 、图片长度、宽度和颜色数。分析非真彩色 bmp 图像结构[2] 如表 1 所示。

表 1　非真彩色 bmp 图像结构

文件头	结构体类型，长度固定 14 字节
信息头	结构体类型，长度固定 40 字节
调色板	4 字节代表一种颜色，其中真彩色图像不需要调色板
位图数据	非真彩色图像中使用到了调色板，数据存储的是调色板的索引值，否则是 RGB 值

分析 bmp 图片各结构体，可得图片关键信息如表 2 所示。

表 2　图片关键信息

名称	偏移量(字节)	长度(字节)	说　明
图像宽度	18	4	位于信息头结构体中
图像高度	22	4	位于信息头结构体中
色彩数	50	4	位于信息头结构体中
调色板	54	4 × 色彩数	1 种颜色需要 4 字节
位图数据	54 + 4 × 色彩数	宽度 × 高度	1 个像素值需要 1 个字节

在内核中访问 I/O 内存之前，需首先使用 ioremap( ) 函数将设备所处的物理地址映射到虚拟地址。ioremap( ) 函数返回一个特殊的虚拟地址，该地址可用来存取特定的物理地址范围。ioremap( ) 函数获得的虚拟地址应该被 iounmap( ) 函数释放[3] 。

以 800 × 480 的 bmp 图像为例，其中 remapped_area 为图像在内存中的首地址，先调用 ioremap( ) 函数映射所需的内核虚拟地址空间，再调用 memcpy( ) ，将内存中的图片信息数据拷贝到图片数组结构中，之后调用 iounmap 函数释放内存空间映射。

```
memcpy( &logo_linux_clut224 - > width, remapped_area + 18,4) ; //图片宽度
memcpy( &logo_linux_clut224 - > height, remapped_area + 22,4) ; //图片高度
memcpy( &logo_linux_clut224 - > clutsize, remapped_area + 50,4) ; //图片颜色数
```

memcpy(logo_linux_clut224 - > clut, remapped_area + 54, logo_linux_clut224 - > clutsize* 4) ; //图片调色板

memcpy(logo_linux_clut224 - > data, remapped_area + logo - > clutsize* 4 + 54, (logo_linux_clut224 - > width) * (logo_linux_clut224 - > height)) ; //图片数据

采用以上方法进行试验，发现显示的图片存在 2 个问题，其一颜色不正确，其二图像发生镜像。

对比 bmp 格式和 ppm 格式图片的调色板和位图数据，bmp 文件调色板中 4 字节代表 1 种颜色，按 BGR 顺序存放[3]，第 4 个字节保留，ppm 文件调色板中 3 字节代表 1 种颜色，按 RGB 顺序存放。bmp 文件位图数据按从下到上，从左到右的方式存储，ppm 文件位图数据按从上到下，从左到右的方式存储，且其值加上 32 后存储在内核图片数组中。

由于内核 logo 文件是 ppm 文件，得到 bmp 图像信息后，要将其转换为 ppm 文件的存储方式。取调色板每种颜色 BGR 分量按 RGB 顺序存储，位图数据加上 32 后改为从上到下，从左到右存储。

4　测试及应用

4.1　测试过程

硬件测试平台是以 ARM S3C2440 为微处理器的嵌入式控制系统，测试步骤为:

a. 在 PC 机上用 Photoshop，将分辨率为 800 × 480 的任意图片，保存为小于 224 色的 bmp 文件。

b. 在 Bootloader 下载模式下，执行命令 run install-kernellogo(tftp 30000000 kernellogo. bmp) ，通过 TFTP 协议将图片下载到 Flash 分区。

c. 改变启动参数，内核启动前先将 bmp 图片读到约定内存。

d. 内核文件中，删除静态 ppm 文件及其生成数组过程，增加从约定内存获取 bmp 图片信息并转换成 ppm 内核图片数组的相关函数，打印出图片相关信息。

e. 如果要更换 logo，只需根据步骤 a 和步骤 b，将 NAND Flash 中的图片文件更换即可，与内核文件无关。

用二进制查看工具 BinView 打开图片并用十六进制显示，对比内核打印的图片信息，可知用上述方案能正确获得图片信息。

4.2　应用

该技术已经应用在 ARM S3C2440 平台下嵌入式 Linux 数控系统中。用该设计方案，将 223 色分辨为 800 ×480 的 bmp 图片显示出来，其结果如图 4 所示，效果良好。

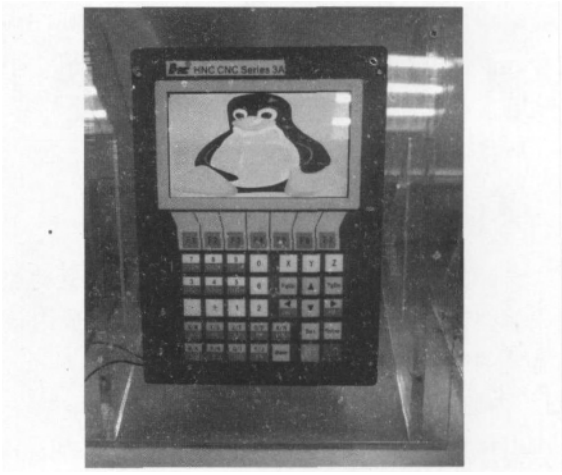

图 4　图片显示效果

5　结束语

介绍了 Linux 操作系统内核 logo 显示原理、更换方法及存在的局限性，对 bmp 文件和 ppm 文件格式进行对比分析，根据 Bootloader 及内核启动相关技术，实现了存放在 NAND Flash 中的 bmp 图片的显示过程，以及内核 logo 图片的快速更换。成功应用在 ARM 平台下嵌入式 Linux 数控系统中。该技术在内核文件不变的情况下，实现了 logo 的快速更换，即更换内核 logo 时不再需要重新编译内核，方便了内核版本的统一，减少了内核设计的烦琐性，加快了产品开发进度。

参考文献:

[1] 韦东山. 嵌入式 Linux 应用开发[M]. 北京: 人民邮电出版社，2008.

[2] 耿　迅. VC 图像处理系列之一——图像读写篇[J]. 电脑编程技巧与维护，2005，(9) : 12 - 16.

[3] 宋宝华. 设备驱动开发详解[M]. 北京: 人民邮电出版社，2010.

作者简介: 赵国伟　(1987 -)，男，山东菏泽人，硕士研究生，研究方向为数控技术和嵌入式数控系统; 唐小琦　(1957 -)，男，湖南邵阳人，博士，教授，博士研究生导师，研究方向为数控技术、现场总线技术等; 宋　宝　(1974 -)，女，湖北黄冈人，博士后，副教授，研究方向为数控技术和嵌入式数控系统等。