基于 VxWorks 下 dosFS 的块设备开发研究

吴 云
（中国航空工业集团公司西安航空计算技术研究所，陕西 西安 710077）

摘要：VxWorks 环境下，介绍了在 dosFs 文件系统下完成对块设备驱动开发过程。首先介绍 dosFs 文件系统的初始化过程，着重分析了块设备内核驱动的结构，详细介绍了设备驱动初始化及驱动主要函数。文章对在 VxWorks 系统与 dosFs 文件系统下开发块设备进行了总结，以便用户可以快捷地对块设备进行配置开发。

关键词：VxWorks；dosFS；块设备
中图分类号：TP316.2 **文献标识码**：A **文章编号**：1673-1131(2017)11-0136-02

0 引言

Vxworks 作为高性能的实时操作系统，在航空、航天等高精尖技术领域，具有广泛应用。块设备主要以“块”来存取各种数据，为了便捷管理各种数据，文件系统将块设备封装为独立的文件，方便数据维护。

VxWorks 为块设备提供了 rawFs 和 dosFs 两种文件系统。rawFs 将整个块存储设备看作一个大文件，连续的现行存储，不具有层次性视图，rawFs 通过 CBIO 中间层与底层设备驱动进行缓冲。dosFs 文件系统通过不同等级设置来管理目录、子目录和包含文件，具有可视化。支持分配连续文件，提高嵌入式设备访问效率，提供了十分灵活的块数据管理方法，与广泛可用的存储器具有兼容性。支持 NFS 系统。从而为块设备的开发提供了很好的支撑。

1 DosFS 文件系统初始化

块设备通过 dosFs 使用之前，dosFs 首先需要完成初始化，这个过程在 VxWorks 的启动过程中完成。

块设备驱动程序与 I/O 系统关系如图 1 所示。

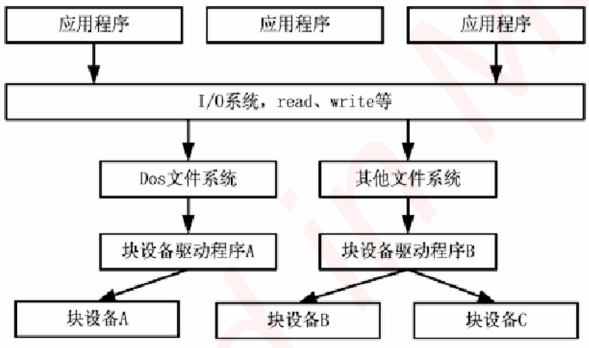

图 1 块设备驱动程序与 I/O 系统的关系

dosFs 一方面对 I/O 子系统进行管理，首先调用 iosDrvInstall 函数注册 dosFs 中间层驱动函数，而 dosFs 中间层通过 dosFsLibInit 函数完成与 I/O 子系统层之间的衔接。通过调用“iosDrvInstall”将系统调用的 creat、delete、open、close、read、write、iocll 等函数注册到驱动程序列表；另一方面调用底层设备驱动，初始化一个 BLK_DEV 结构，获取必须的底层驱动信息。iosDrvInstall 的函数原型如下。

int iosDrvInstall(FUNCPTR pCreate, FUNCPTR pDelete, FUNCPTR pOpen, FUNCPTR pClose , FUNCPTR pRead , FUNCPTR pWrite , FUNCPTR pIoctl)

其次，用户不能直接对底层设备发送请求进行操作，而是通过设备驱动表找到对应的注册函数，通过这些函数进行响应作为过渡，最终将通过底层驱动进行操作。dosFs 通过 dosFslib 库完成与底层设备驱动之间的连接，其中的核心函数是 dosFsDevInit。

```
STATUS  dosFsDevCreate
{

char *  pDevName,   /*设备名*/
    CBIO_DEV_ID   cbio, /*CBIO 设备*/
    u_int   maxFiles,    /*最大同时打开文件数*/
    u_int   autoChkLevel  /*文件一致性检查处理级别*/
    };
```

需要注意的是 CBIO_DEV_ID 实际上是一个 cbioDev 结构指针，并且实际上总是调用 BLK_DEV 结构指针，将其封装成 CBIO 结构。

最后调用 dosFsVolFormat 函数完成格式化操作，以便正确地访问块设备。

```
STATUS  dosFsVolFormat
{

void *  device,     /*被格式化的块设备节点名*/
    int  opt,          /*格式化选项*/
    FUNCPTR  pPromptFunc /*格式化参数*/
    }
```

一般来说对已小于 2G 的块设备采用 FAT16 进行格式化，大于 2G 的设备采用 FAT32 进行格式化。

2 块设备驱动内核结构

文件系统层需要与底层设备驱动进行交互，因而定义了 BLK_DEV 内核结构作为交互的媒介。BLK_DEV 的结构定义如下：

```
typedef struct
{
FUNCPTR  bd_blkRd; /*块设备数据块读取函数*/
FUNCPTR  bd_blkWrt; /*块设备数据块写入函数*/
FUNCPTR  bd_ioctl /*块设备控制函数*/
FUNCPTR  bd_reset /*复位设备函数*/
FUNCPTR  bd_statusChk /*查询设备状态函数*/
FUNCPTR  bd_statusChk /*查询设备状态函数*/
```

```
BOOL  bd_removable /*可移动标志*/
ULONG bd_nBlocks;  /*设备总块数*/
ULONG bd_bytesPerBlk; /*每块字节数*/
ULONG bd_blksPerTrack; /*每磁道块数*/
ULONG bd_nHeads; /*磁头数*/
Int bd_retry; /*I/O 错误时尝试次数*/
Int bd_mode; /*操作模式*/
BOOL bd_readychanged; /*设备状态*/
}
```

上述结构中，读写函数是其中的核心功能。块设备中不能进行单字节的读写，只能每次以一个块的方式进行读写，文件系统将完成目录或文件向块的转换过程。

一般情况下会将内核信息与驱动信息定义为一个整体方便同时访问。驱动自定义结构如下。

```
Typedef struct _blk_dev_struct
{
BLK_DEV blkdev;
UINT32 regBase; /*硬盘控制寄存器基地址*/
FUNCPTR intHandler; /*硬盘控制器中断响应函数*/
UINT8 intLevel; /*硬盘控制器中断号*/
}
```

在块设备驱动中既可以采用轮询的方式进行读写，也可以采用中断的方式进行读写，具体由项目的读写要求和硬件设计来进行设计。

3 块设备驱动

一般的块设备数据传输有两种模式，PIO 模式和 DMA 模式。PIO 模式是一种通过 CPU 通过 I/O 端口指令来进行读写的数据交换模式，传输速度从 3.3MB/s 到 16.6MB/s 不等。DMA 是直接访问内存进行读取数据，CPU 只需要向 DMA 控制器下达指令，让 DMA 自行完成数据传输，完成后向 CPU 反馈信息，其最大传输速度也是 16.6MB/S，但是极大地减轻了 CPU 的负荷。但是无论哪种传输方式，对块设备的底层驱动是一致的。

块设备首先要进行初始化过程。通过 ataBlkDrv 函数完成底层设备控制器寄存器的配置、校正磁头、复位块设备等工作。如果是多个块设备，则需要分别调用一次 ataBlkDrv 函数，同时给出块设备控制编号、块设备控制器寄存器基地址、中断号。ataBlkDrv 实际上尚未完成对 BLK_DEV 提供给上层使用的关键结构成员的初始化。这个工作交由 ataBlkDevCreate 函数完成。

ataBlkDevCreate 完成驱动自定义结构的分配，初始化以及底层所需要的其他资源的分配工作。最重要的是，该函数将 BLK_DEV 作为函数返回值，提供给 CBIO 中间层使用，所以上层用户对底层的读写、控制、状态查询、复位等都通过 BLK_DEV 结构信息完成。

块设备驱动中读写函数名无关紧要，因为他们总是要调用函数 ataBlkRw。

ataBlkRw

```
{
BLK_DEV *pDev, /*块设备结构指针*/
Int startBlk, /*起始块号*/
Int nBlks, /*需要读取的总块数*/
char *pBuf, /*存放被读取数据的缓冲区*/
int bd_mode
};
```

如果是写，则 bd_mode 设置为 O_RDONLY，如果是读，则 bd_mode 设置为 O_WDONLY。其他参数需要调用的读写函数给出。

接口复位函数主要是在硬件产生问题后复位硬件。其过程如下：

```
STATUS ataReset(ATA_BLK_DEV *pDev)
{
int retrySeek=0;
int status=OK;
while(ataCmd(pDev,ATA_CMD_RESET,0,0)！=OK)

{
    If (++retrySeek>ataRetry)
    {
      Status=ERROR;
}
}
return status;
}
```

状态接口函数的主要作用是检查当前磁盘是否发生变化。对于可更换存储介质的存储设备来说，这个接口非常关键。

至此，完成了对文件系统的初始化，对块设备驱动的设计，并通过 BLK_DEV 内核设计完成了两者之间的交互。上层可以通过直接调用驱动完成对底层的操作。

4 结语

本文首先从文件系统层入手，介绍了初始化过程，使得块设备对上层可见，从块设备内核层次出发介绍了不同层次之间的衔接关系，最后给出了设备驱动的主要函数和介绍。按照本文流程，可以快速实现块设备驱动配置。

参考文献：

[1] 高斌, 翟江涛, 薛朋骏. 一种 VxWorks 文件系统层访问控制方法[J]. 江苏科技大学学报(自然科学版), 2015, 29(5): 462-466.

[2] 马昕, 张士洋. 基于Vxworks的大容量信息存储实现方法研究[J]. 网络新媒体技术, 2007, 28(10):1102-1106.

[3] 韩林. 基于 Vxworks 的嵌入式记录回放系统[D]. 电子科技大学, 2011.

作者简介:吴云(1990-)，男，硕士，助理工程师，软件可靠性。