文件系统

ZJUNIX

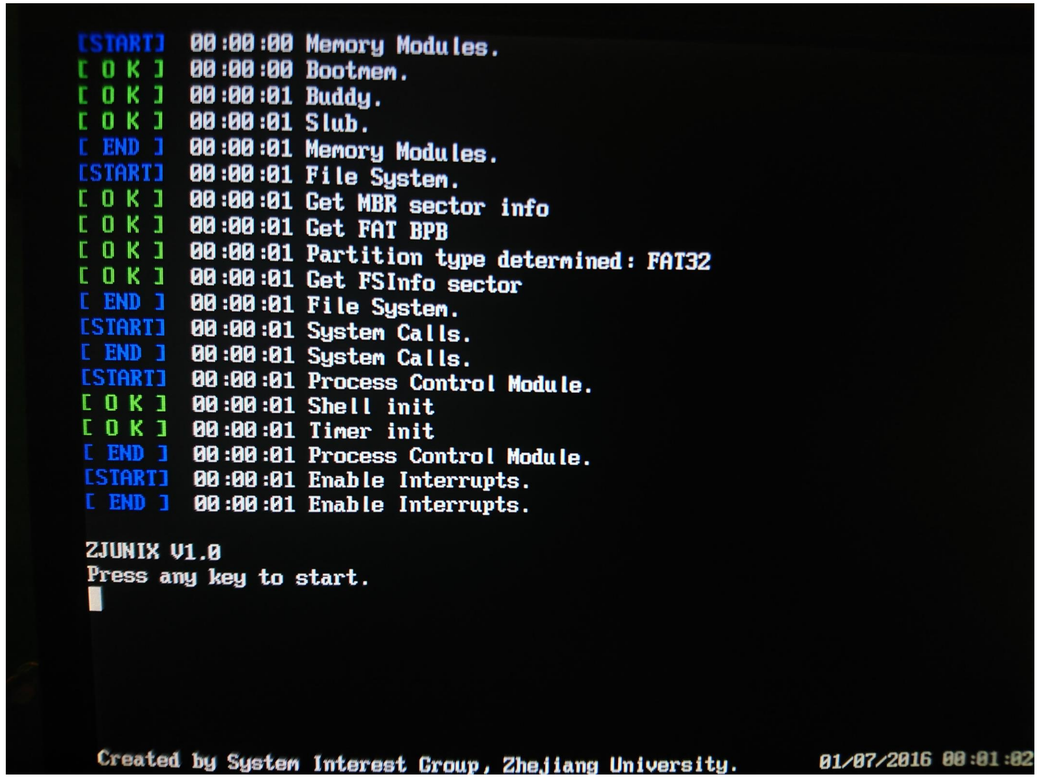
目录

文件系统简介

- 作用
- 种类
- 难点

文件系统的作用

- 保存和管理文件
- 附加的其他功能
  - 日志记录
  - 安全性保护
- **保存系统启动文件**
  - 在启动时载入系统镜像
  - 替换系统镜像来升级



文件系统种类

- 个人电脑使用的文件系统大多比较常见
  - FAT, NTFS, EXT
- 操作系统一般会提供类似VFS(虚拟文件系统)的机制来降低文件系统读写的难度
- 对于我们的操作系统，我们希望实现一个通用而且简单的操作系统，不需要很复杂，但是可用且好用

文件系统类型	常见格式
磁盘文件系统	FAT, NTFS, EXT …
闪存文件系统	
磁带文件系统	
网络文件系统	NFS, Samba

来源：https://en.wikipedia.org/wiki/File_system#Types_of_file_systems

文件系统实现

- 在操作系统中，文件系统的原理是相对简单的
  - 操作逻辑比较简单
  - 数据结构比较清晰
  - 一般具有面向对象的特性，便于开发和理解
- 对比file结构体与task_struct结构体(Linux v4.12.5)
  - file: 包含19项数据
  - task_struct: 包含约218项数据
  - 相对而言文件系统的各个数据结构比较紧凑，操作也比较简单，所以在操作系统中属于原理相对简单的部分

文件系统实现的难点

- 但是文件系统实现的难点在于
  - 测试困难：
    - 一旦写错代码，有可能造成数据损坏，导致全盘格式化并重装系统
  - 开发困难：
    - 文件系统操作的特殊情况比较多，完全覆盖并进行测试需要大量时间
  - 标准繁多：
    - 不同的文件系统有其自己的标准，完全遵守才能有较好的通用性

我们将要设计的文件系统

- FAT32
  - 遵循主要标准
    - ZJUNIX上可读写的文件，在其他操作系统上也可以读写
  - 4KB簇大小
    - 便于开发和调试
    - 可以格式化32GB以下的分区
  - 不处理长文件名
    - FAT32的长文件名比较难处理
    - 只处理常见的8-3格式文件名

设计与实现

- 文件系统实现
- 缓冲区实现

文件控制块

- 记录文件信息
  - 路径
  - 当前读写指针位置
  - 目录项在磁盘上的位置
  - 文件数据缓冲

```c
/* file struct */
typedef struct fat_file {
    unsigned char path[256];
    /* Current file pointer */
    unsigned long loc;
    /* Current directory entry position */
    unsigned long dir_entry_pos;
    unsigned long dir_entry_sector;
    /* current directory entry */
    union dir_entry entry;
    /* Buffer clock head */
    unsigned long clock_head;
    /* For normal FAT32, cluster size is 4k */
    BUF_4K data_buf[LOCAL_DATA_BUF_NUM];
} FILE;
```

目录项

- 目录项是什么
  - 在FAT32中，文件的属性与数据是分开存储的
  - 一个文件/目录的属性信息就是目录项
  - 同一目录下的文件/目录的目录项都紧凑地储存在一起
- 目录项的结构
  - FAT32中一个目录项为32字节
  - 包括文件名，创建时间，数据存储位置等信息

目录项

32字节 原始数据

```
union dir_entry {
    u8 data[32];
    struct dir_entry_attr attr;
};
```

```
struct __attribute__((__packed__)) dir_entry_attr {
    u8 name[8];
    u8 ext[3];
    u8 attr;
    u8 lcase;
    u8 ctime_cs;
    u16 ctime;
    u16 cdate;
    u16 adate;
    u16 starthi;
    u16 time;
    u16 date;
    u16 startlow;
    u32 size;
};
```

原始数据对应的属性

目录项

```
union dir_entry {
u8 data[32];
struct dir_entry_attr attr;
};
```

按字节存储的原始数据

限于篇幅仅列出16字节，其余信息请参考FAT32标准

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Name								Ext			Attr	Lcase	Ctime	Ctime	

按照不同的数据格式访问，如字符串、short、int格式

向data中读入原始数据 → 在attr中以特定数据结构操作数据 → data与attr共享内存，同步修改 → 从data中读出原始数据，写入磁盘

ZJUNIX

FAT32格式

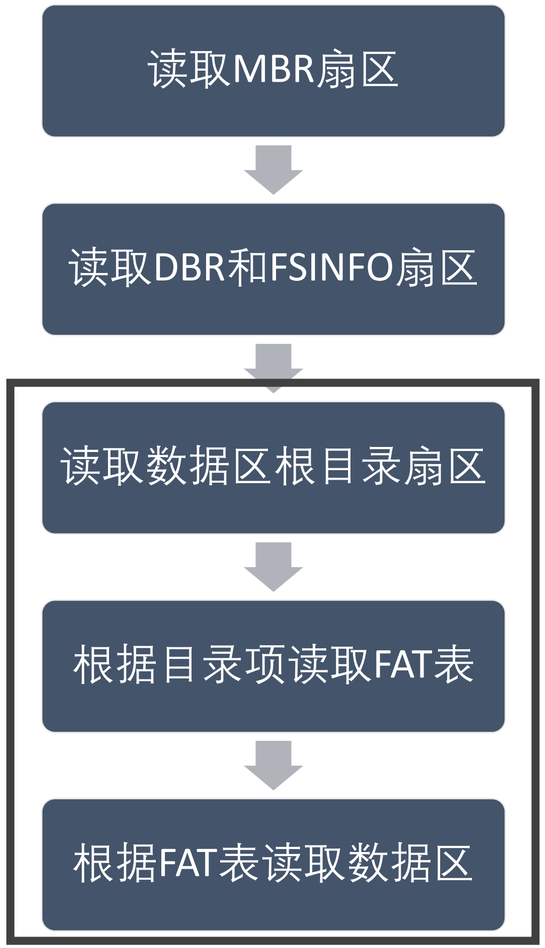
	保留区1		保留区2			文件分配表		数据区
结构	MBR (512字节)		DBR (512字节)	FSINFO (512字节)		FAT1	FAT2	存放实际数据
物理地址范围	0~N-1扇区		N~N+(M-1)扇区			N+M~N+M+(K-1)	N+M+K~N+M+M+(K-1)	N+M+K+K…
说明	1. N(第一个保留区大小): MBR表中从0x1C6开始的四个字节		1. M (第二个保留区大小): DBR中从0x0E开始的两个字节			1. K (FAT表大小): DBR中从0x24开始的四个字节		

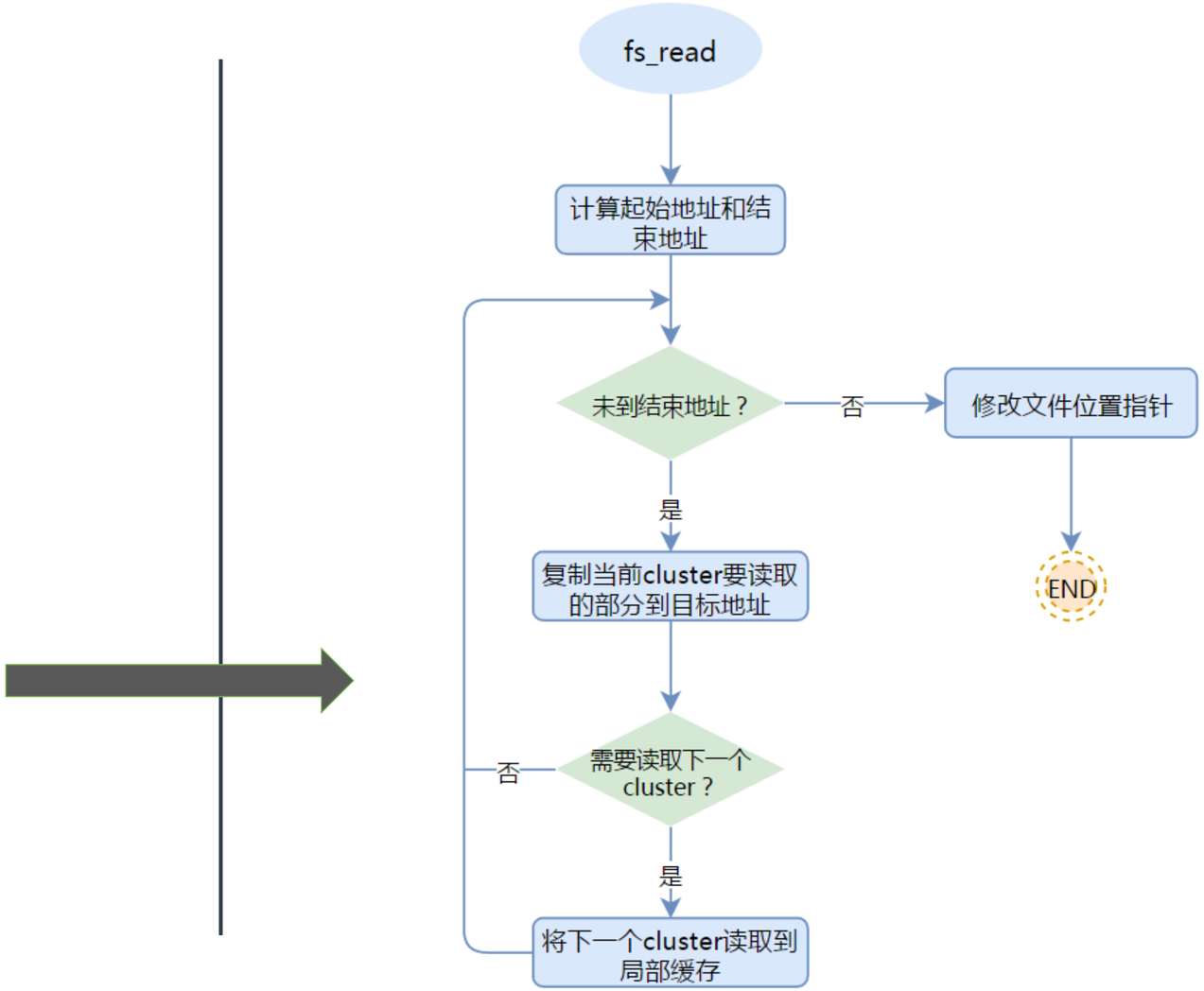
读取MBR扇区 → 读取DBR和FSINFO扇区 → 读取数据区根目录扇区 → 根据目录项读取FAT表 → 根据FAT表读取数据区

文件系统初始化


```
/* Init bufs */
kernel_memset(meta_buf, 0, sizeof(meta_buf));
kernel_memset(&fat_info, 0, sizeof(struct fs_info));

/* Get MBR sector */
if (read_block(meta_buf, 0, 1) == 1)
    goto init_fat_info_err;


log(LOG_OK, "Get MBR sector info");
fat_info.base_addr = get_u32(meta_buf + 446 + 8);

/* Get FAT BPB */
if (read_block(fat_info.BPB.data, fat_info.base_addr, 1) == 1)
    goto init_fat_info_err;
log(LOG_OK, "Get FAT BPB");
```

读取流程


读取MBR扇区
读取DBR和FSINFO扇区
读取数据区根目录扇区
根据目录项读取FAT表
根据FAT表读取数据区

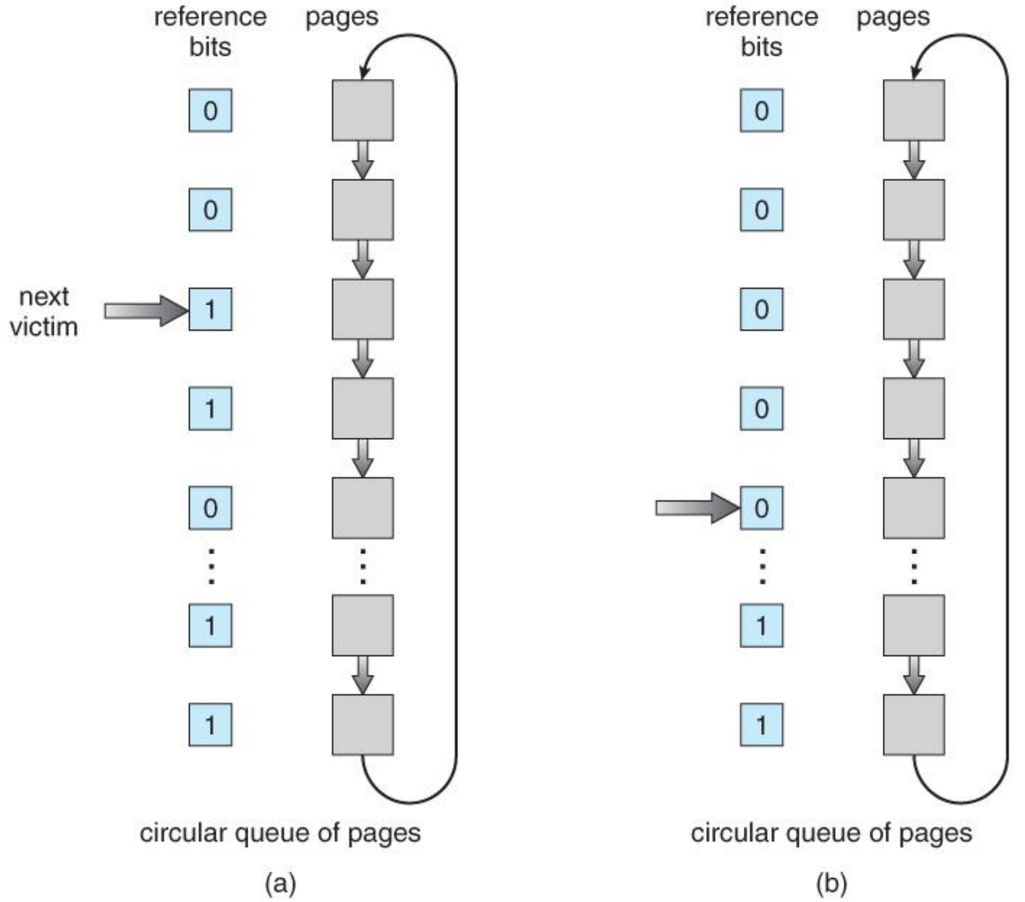

fs_read
计算起始地址和结束地址
未到结束地址？
否
修改文件位置指针
是
复制当前cluster要读取的部分到目标地址
END
需要读取下一个cluster？
否
是
将下一个cluster读取到局部缓存

文件读入后……

- 一个文件已经被找到了，但接下来还面临这些问题：
  - 文件内容应该如何保存在内存里
  - 需要开辟的缓冲区有多大
  - 缓冲区满了怎么办
- 所以现在需要一个合适的缓冲区管理算法
  - 可以参考内存的缓冲区管理

缓冲区管理

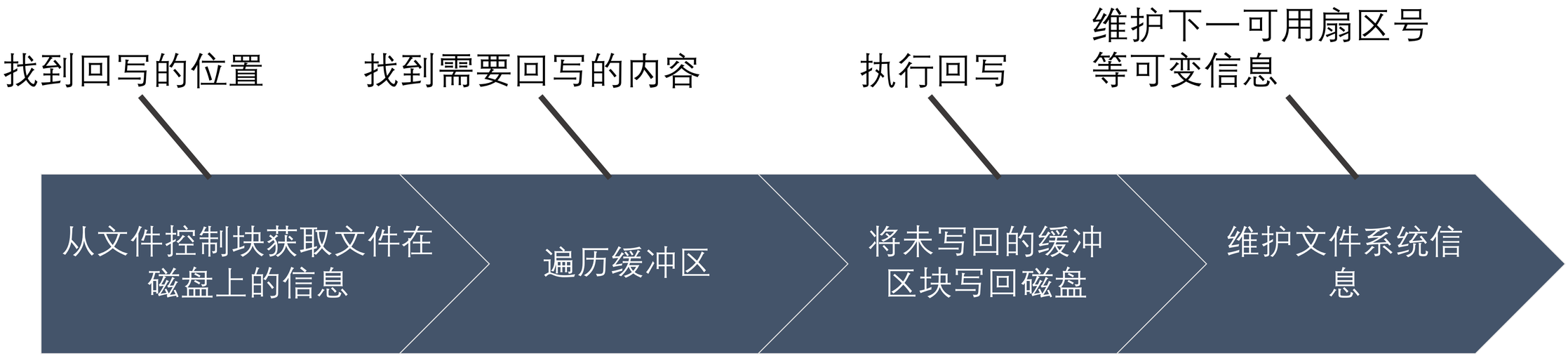
- second chance 算法
  - ref bit为1代表此块最近被用过
  - 第一轮查找
    - 发现ref bit为0的块，则选中
    - 对于ref bit为1的快，设置ref bit为0
  - 第二轮查找
    - 如果第一轮没有任何块被选中，则进入第二轮
    - 发现第一个ref bit为0的块，则选中



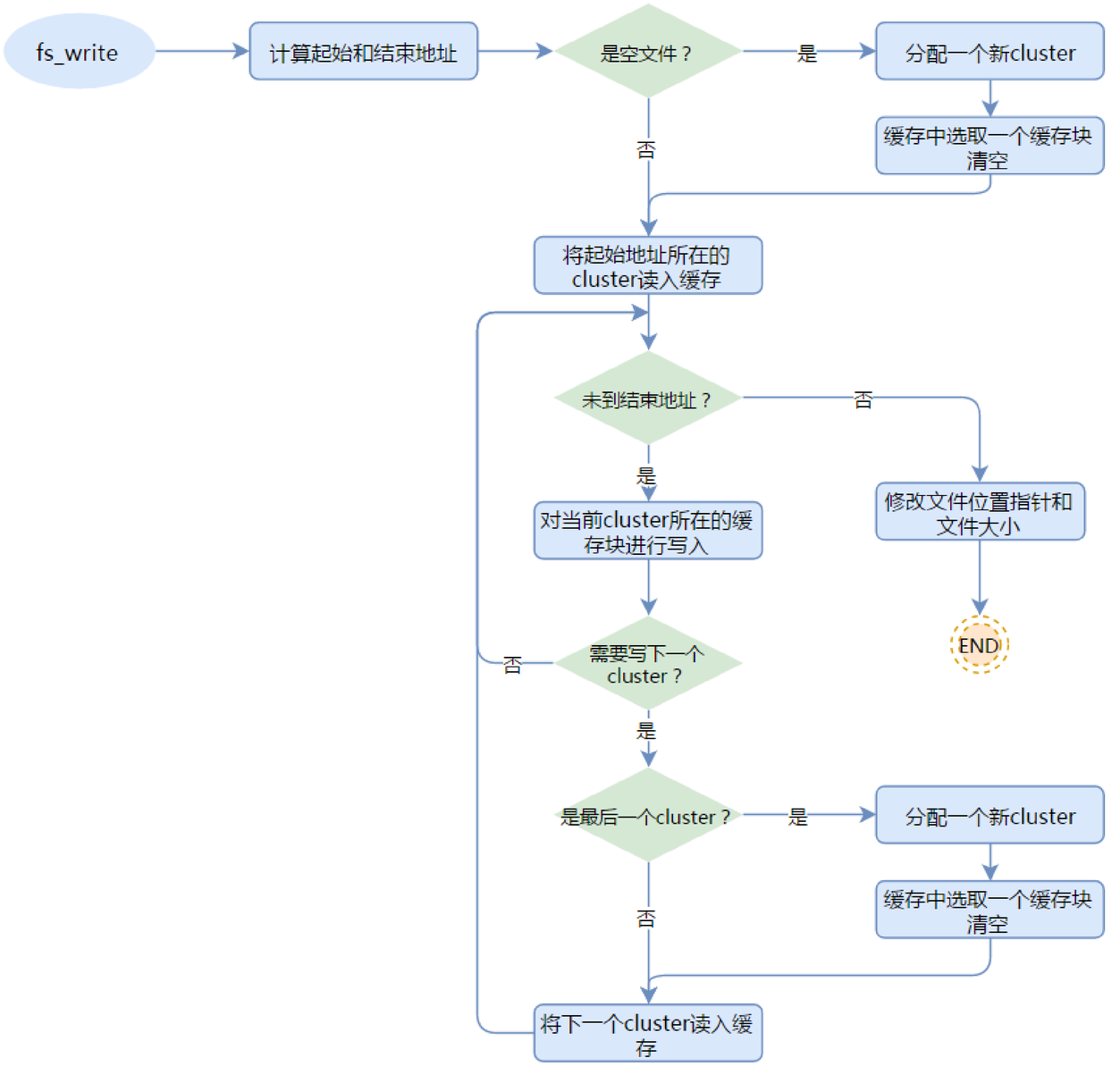
缓冲区管理

- second chance 算法避免了重要的块被换出
  - 这个算法在内存管理中也有应用场景
  - 只是相对地避免了重要的块被换出
- 也可以考虑使用其他管理算法
  - 先入先出（FIFO）
  - 最近最少使用（LRU）
- 与缓冲区管理相关的代码都在kernel/fs/fscache中，可以替换不同的管理算法

文件回写


找到回写的位置
找到需要回写的内容
执行回写
维护下一可用扇区号
等可变信息
从文件控制块获取文件在磁盘上的信息
遍历缓冲区
将未写回的缓冲区块写回磁盘
维护文件系统信息

文件回写


fs_write
计算起始和结束地址
是空文件？
是
分配一个新cluster
缓存中选取一个缓存块
清空
否
将起始地址所在的
cluster读入缓存
未到结束地址？
否
是
对当前cluster所在的缓
存块进行写入
修改文件位置指针和
文件大小
END
需要写下一个
cluster？
否
是
是最后一个cluster？
是
分配一个新cluster
缓存中选取一个缓存块
清空
否
将下一个cluster读入缓
存

THANK YOU