

Lab14 文件系统1

一、实验概述

本周我们会对文件系统fat和unix文件系统进行简单介绍。并对ucore代码使用的Simple File System结构进行介绍。

二、实验目的

1. 了解fat文件系统结构
2. 了解unix文件系统的inode
3. 了解Simple File System结构

三、实验项目整体框架概述

Lab14

```
|— kern
|   |— debug
|   |— driver
|   |   |— clock.c
|   |   |— clock.h
|   |   |— console.c
|   |   |— console.h
|   |   |— ide.c //设备初始化
|   |   |— ide.h
|   |   |— intr.c
|   |   |— intr.h
|   |   |— kbdreg.h
|   |   |— ramdisk.c //磁盘初始化及相关函数
|   |   |— ramdisk.h
|   |— fs //文件系统
|   |   |— devs
|   |   |   |— dev.c
|   |   |   |— dev_disk0.c
|   |   |   |— dev.h
|   |   |   |— dev_stdin.c
|   |   |   |— dev_stdout.c
|   |   |— file.c
|   |   |— file.h
|   |   |— fs.c
|   |   |— fs.h
|   |   |— iobuf.c
|   |   |— iobuf.h
|   |   |— sfs
|   |   |   |— bitmap.c
|   |   |   |— bitmap.h
|   |   |   |— sfs.c
|   |   |   |— sfs_fs.c
```

```

| | | |— sfs.h
| | | |— sfs_inode.c
| | | |— sfs_io.c
| | | |— sfs_lock.c
| | |— swap
| | | |— swapfs.c
| | | |— swapfs.h
| | |— sysfile.c
| | |— sysfile.h
| | |— vfs
| | | |— inode.c
| | | |— inode.h
| | | |— vfs.c
| | | |— vfsdev.c
| | | |— vfsfile.c
| | | |— vfs.h
| | | |— vfslookup.c
| | | |— vfspath.c
| |— init
| |— libs
| |— mm
| |— process
| |— schedule
| |— sync
| |— syscall
| |— trap
|— libs
|— Makefile
|— tools
| |— function.mk
| |— kernel.ld
| |— mksfs.c //制作镜像
| |— user.ld
|— user //用户文件

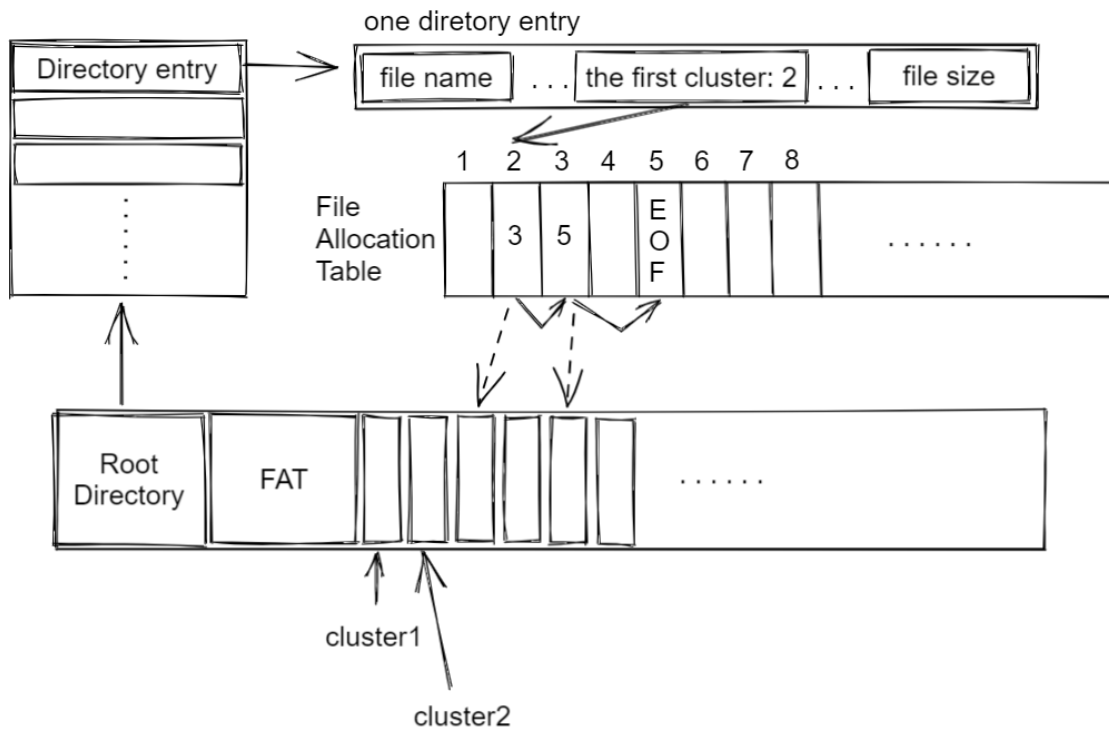
```

四、实验内容

1. File Allocation Table

FAT文件系统采取链式存储结构，将磁盘分为若干簇（cluster），簇的大小是固定的。文件以簇为单位进行分割存储在磁盘上不一定连续的簇中。

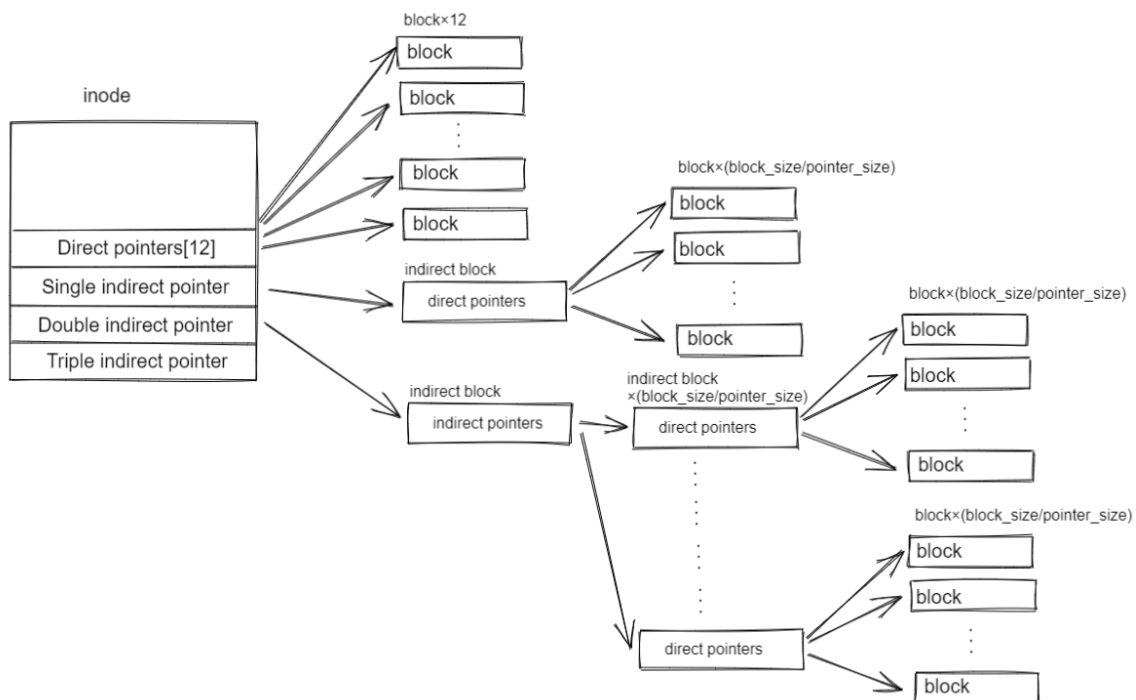
磁盘中存在一个Root Directory区域，其中存储了若干目录项，一个目录项包含了一个文件的信息，通过目录项中的起始簇号可以在数据区中找到文件的第一个簇，并可以根据起始簇号通过查询File Allocation Table找到第二个簇的簇号，依次类推知道找到文件的最后一个簇。



2. Unix inode

UNIX操作系统使用inode结构体用于管理文件。每个文件（文件夹）都有自己的inode，inode中包含了文件（文件夹）的各类信息，每个inode拥有唯一的inode号。

unix文件系统被分割成固定大小的块（block）。文件inode中包含的12直接指针和3个间接指针指向了文件的数据块。下图表示了直接指针和间接指针的链接方式，直接指针直接指向存储数据的block，而间接指针指向存储指针的block，每个block则可以存储 $\text{block_size}/\text{指针大小}$ 个指针。

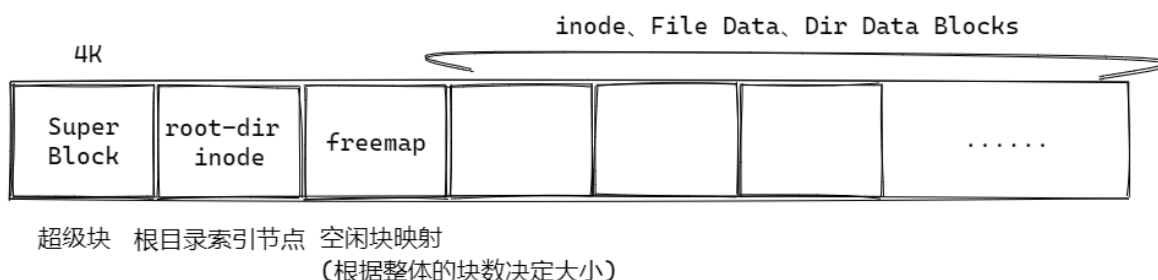


UNIX中，文件夹和文件都是文件，它们都有自己的inode，inode中根据文件的信息可以区分当前文件是否为文件夹。如果是文件夹inode则该inode的直接指针指向的数据块是一个Directory Block，里面会包含该目录下的文件的inode号；如果是文件inode则该inode的指针们指向的数据块都是Data Block，用于存储文件数据。

3. Simple File System简单文件系统

在本次实验中，我们实现了一块符合简单文件系统标准的磁盘镜像 `sfs.img`。磁盘的使用是以扇区（Sector）为单位的，为了实现简单，我们设置磁盘扇区大小为4KB。

下面我们介绍简单文件系统SFS：

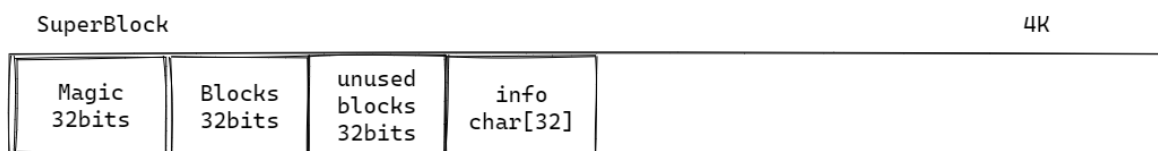


超级块

第0个块（4K）是超级块（superblock），它包含了关于文件系统的所有关键参数，当计算机被启动或文件系统被首次接触时，超级块的内容就会被装入内存。其定义如下：

```
struct sfs_super {  
    uint32_t magic;           /* magic number, should be SFS_MAGIC */  
    uint32_t blocks;          /* # of blocks in fs */  
    uint32_t unused_blocks;   /* # of unused blocks in fs */  
    char info[SFS_MAX_INFO_LEN + 1]; /* information for sfs */  
};
```

超级块包含一个成员变量magic，其值为0x2f8dbe2a，内核通过它来检查磁盘镜像是否是合法的 SFS img；成员变量blocks记录了SFS中所有block的数量，block的数量乘以block的大小也就是 img 的大小；成员变量unused_block记录了SFS中还没有被使用的block的数量；成员变量info包含了字符串"simple file system"。



根inode

第1个块放了一个root-dir的inode，用来记录根目录的相关信息。这里只要理解root-dir是SFS文件系统的根结点，通过这个root-dir的inode信息就可以定位并查找到根目录下的所有文件信息。

Freemap

从第2个块开始，根据SFS中所有块的数量，用1个bit来表示一个块的占用和未被占用的情况。这个区域称为SFS的freemap区域，这将占用若干个块空间。

剩余块

在剩余的磁盘空间中，存放了所有其他目录和文件的inode信息和内容数据信息。需要注意的是虽然inode的大小小于一个块的大小（4096B），但为了实现简单，每个inode都占用一个完整的block。

磁盘上的inode和entry

```
/*inode (on disk)*/
struct sfs_disk_inode {
    uint32_t size;                //如果inode表示常规文件，则size是文件大小
    uint16_t type;                //inode的文件类型
    uint16_t nlinks;              //此inode的硬链接数
    uint32_t blocks;              //此inode的数据块数的个数
    uint32_t direct[SFS_NDIRECT]; //此inode的直接数据块索引值（有SFS_NDIRECT个）
    uint32_t indirect;            //此inode的一级间接数据块索引值
};
```

SFS 中的磁盘索引节点代表了一个实际位于磁盘上的文件。

如果 inode 表示的是文件，则成员变量 direct[] 直接指向了保存文件内容数据的数据块索引值。indirect 间接指向了保存文件内容数据的数据块，indirect 指向的是间接数据块（indirect block），此数据块实际存放的全部是数据块索引，这些数据块索引指向的数据块才被用来存放文件内容数据。

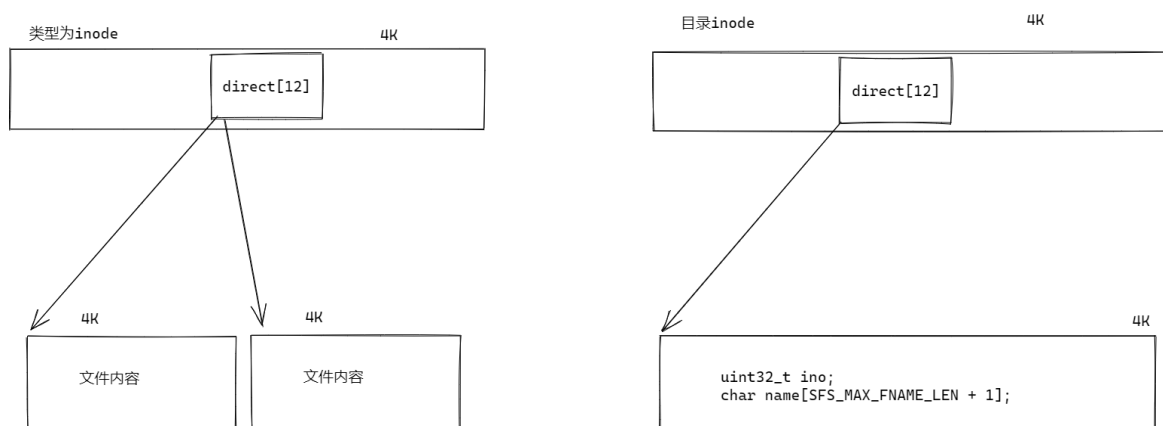
默认ucore 里 SFS_NDIRECT 是 12，即直接索引的数据页大小为 $12 \times 4k = 48k$ 。数据索引表内，0 表示一个无效的索引，inode 里 blocks 表示该文件或者目录占用的磁盘的 block 的个数。indirect 为 0 时，表示不使用一级索引块。

对于普通文件，索引值指向的 block 中保存的是文件中的数据。而对于目录，索引值指向的数据保存的是目录下的文件项，每个文件项占一个Block，文件项内存储了指向文件inode的索引值以及文件名。数据结构如下：

```
/* file entry (on disk) */
struct sfs_disk_entry {
    uint32_t ino;                //对应文件的inode索引
    char name[SFS_MAX_FNAME_LEN + 1]; //文件名
};
```

操作系统中，每个文件系统下的 inode 都应该分配唯一的 inode 编号。SFS 下，为了实现的简便，每个 inode 直接用他所在的磁盘 block 的编号作为 inode 编号。比如，root block 的 inode 编号为 1；每个 sfs_disk_entry 数据结构中，name 表示目录下文件或文件夹的名称，ino 表示磁盘 block 编号，通过读取该 block 的数据，能够得到相应的文件或文件夹的 inode。ino 为 0 时，表示一个无效的 entry。

和 inode 相似，每个 sfs_disk_entry 也占用一个 block。



五、本节知识点回顾

在本次实验中，你需要了解以下知识点：

1. Fat文件系统的文件存储方式
2. inode的作用
3. Simple File System文件系统的构造

七、下一实验简单介绍

下一次实验，我们将具体一点介绍Simple File System。