文件系统概要设计

程子霖

因为不存在分别挂载，所以不存在二级目录一说？

而且，好像硬链接的部分来不及做了……

首先，100MB 的空间，1KB 的数据块大小

1024*100=102400，int 占 4bytes，足够存储这个范围的数

基础结构还是 superblock 占一个 datablock，然后 datablock_bitmap 需要 102400/8*1024=12.5 进位到 13 块

INODE 的 Bitmap 的大小：102400/16*1024=6.25 个 blk，保险起见用 7 个，Inode 个数 7*1024=7168（假设 16KB 一个文件）

一个 inode 占 512B，则 102400/16*2=3200 个块，作为 Inode Table，约 3.2MB?

MARCO

```
#define BLK_SIZE 1024   //一个 block 的大小
#define BLK_MAX_NUMBER 102400    //block 的数量
#define SUPER_BLK_SIZE 1          //超级块的大小
#define DATABLK_BITMAP_SIZE 13      //datablock bitmap 的所占的 block 的个数
#define INODE_BITMAP_SIZE 7         //inode bitmap 所占的 block 的个数
#define INODE_TABLE_SIZE 3200       //inode table  所占块数
#define INODE_SIZE 512              //一个 INODE 的大小
#define FILE_MAX_BLKS 119           //单个文件最大大小，119 个块，482KB（为凑 inode 一个占 512B）
#define DIR_SUB_MAX  20             //一个目录最多有的子项的个数
#define FILE_MAX_NAME 30
struct superblock{
        int  ROOTDIR;       //下面是 superblock 的结构体
        int DATABLK_BITMAP_START;
        int INODE_BITMAP_START;
        int INODE_TABLE_START;
        int DATABLK_START;
        int INODE_REMAIN;
        int DATABLK_REMAIN;
        int ROOTDIR_INODE
}
struct inode{
    char i_FILENAME[FILE_MAX_NAME];   //占 30B 的名称
    Int flag;                         //标志位，0 为目录，1 为文件
    int i_DATABLK[FILE_MAX_BLKS];     //占 476B
}
struct in_bit_map{          //目录文件内的内容，标识目录内文件和目录的使用情况的
    char SUB_USAGE[DIR_SUB_MAX]; //使用一个 char 表示一个目录项有没有被占用。
}
struct initem  {    //目录文件内容，标识目录里的文件，以及他们的 inode 号
    char SUB_NAME[FILE_MAX_NAME];   //存储子项的名字，30B
    int SUB_INO_NO;                 //子项的 inode 号，4B
    int trick[4];                   //凑够一个目录项 50B
```

```
}
```