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

}