文件系统

一 、目录（文件）和文件

1文件的属性

获取一个文件的具体信息函数主要是获得stat结构体

int stat(const char \*pathname, struct stat \*statbuf);

int fstat(int fd, struct stat \*statbuf);

int lstat(const char \*pathname, struct stat \*statbuf);

note:lstat和stat在链接文件的处理上有区别，lstat获取的stat信息是链接文件的属性，stat获取到的信息是链接文件所指向文件的属性。

stat 结构体

All of these system calls return a stat structure, which contains the following fields:

struct stat {

dev\_t st\_dev; /\* ID of device containing file 包含这个文件的设备ID号 \*/

ino\_t st\_ino; /\* Inode number \*/

mode\_t st\_mode; /\* File type and mode 权限信息和文件类型 \*/

nlink\_t st\_nlink; /\* Number of hard links \*/

uid\_t st\_uid; /\* User ID of owner \*/

gid\_t st\_gid; /\* Group ID of owner \*/

dev\_t st\_rdev; /\* Device ID (if special file) 设备号\*/

off\_t st\_size; /\* Total size, in bytes 理论大小\*/

blksize\_t st\_blksize; /\* Block size for filesystem I/O \*/

blkcnt\_t st\_blocks; /\* Number of 512B blocks allocated 实际大小\*/

/\* Since Linux 2.6, the kernel supports nanosecond

precision for the following timestamp fields.

For the details before Linux 2.6, see NOTES. \*/

struct timespec st\_atim; /\* Time of last access 最后一次读的时间 \*/

struct timespec st\_mtim; /\* Time of last modification 最后一次写的时间\*/

struct timespec st\_ctim; /\* Time of last status change 亚数据修改的时间\*/

#define st\_atime st\_atim.tv\_sec /\* Backward compatibility \*/

#define st\_mtime st\_mtim.tv\_sec

#define st\_ctime st\_ctim.tv\_sec

};

note:系统中的*/ etc /passwd记录了对应的用户号和用户名,/ etc / group文件记录了对应的组号和组名*

2文件的权限（读写）

stat结构体中的mode\_t st\_mode成员记录了文件的权限信息和文件类型是一个 16 位整型数

st\_mode:(16 位位图)

1111 1 111 1111 1111 1111

文件类型 粘住位/U-id/G-id owner权限 group权限 other用户权限

文件类型

d 目录

c 字符设备文件

b 块设备文件

- 常规文件

l 符号链接文件

s socket文件

p 管道文件

判断文件类型（宏）:

S\_ISREG(m) is it a regular file?

S\_ISDIR(m) directory?

S\_ISCHR(m) character device?

S\_ISBLK(m) block device?

S\_ISFIFO(m) FIFO (named pipe)?

S\_ISLNK(m) symbolic link? (Not in POSIX.1-1996.)

S\_ISSOCK(m) socket? (Not in POSIX.1-1996.)

权限的更改

int chmod(const char \*pathname, mode\_t mode);

int fchmod(int fd, mode\_t mode);

3 Umask

防止产生权限过松的文件，新建文件的权限与umask有关

4粘住位（t位）

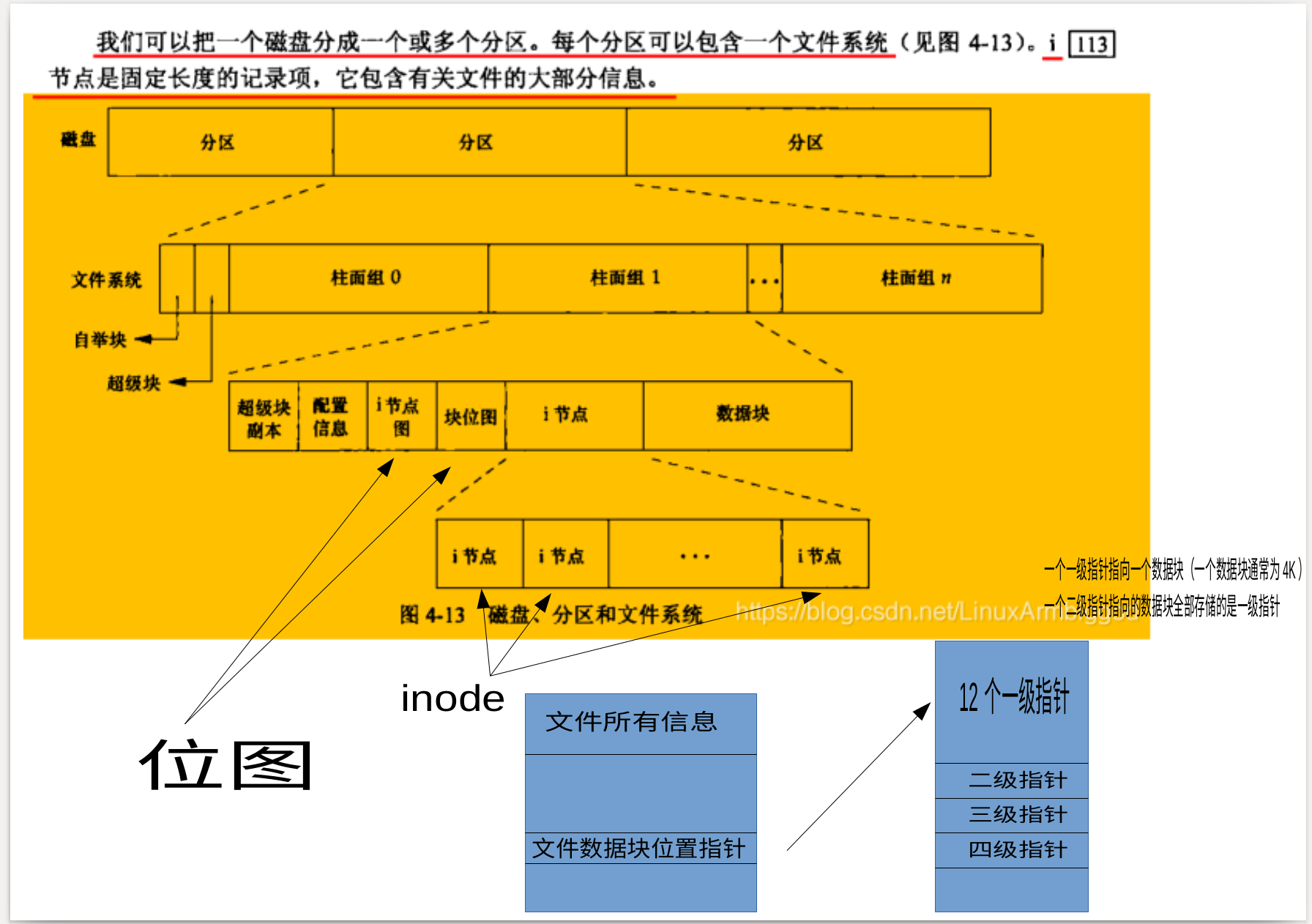
最初用于二进制可执行文件，设置t位，会在内存中为该程序保留数据，加快下次运行的加载速度。现在因为chche的出现，不再需要此位加快运行速度。如今，t位一般设置在/tmp目录，不同用户对该目录的操作会被记录下来。

5文件系统 FAT ，UFS （unix早期文件系统）

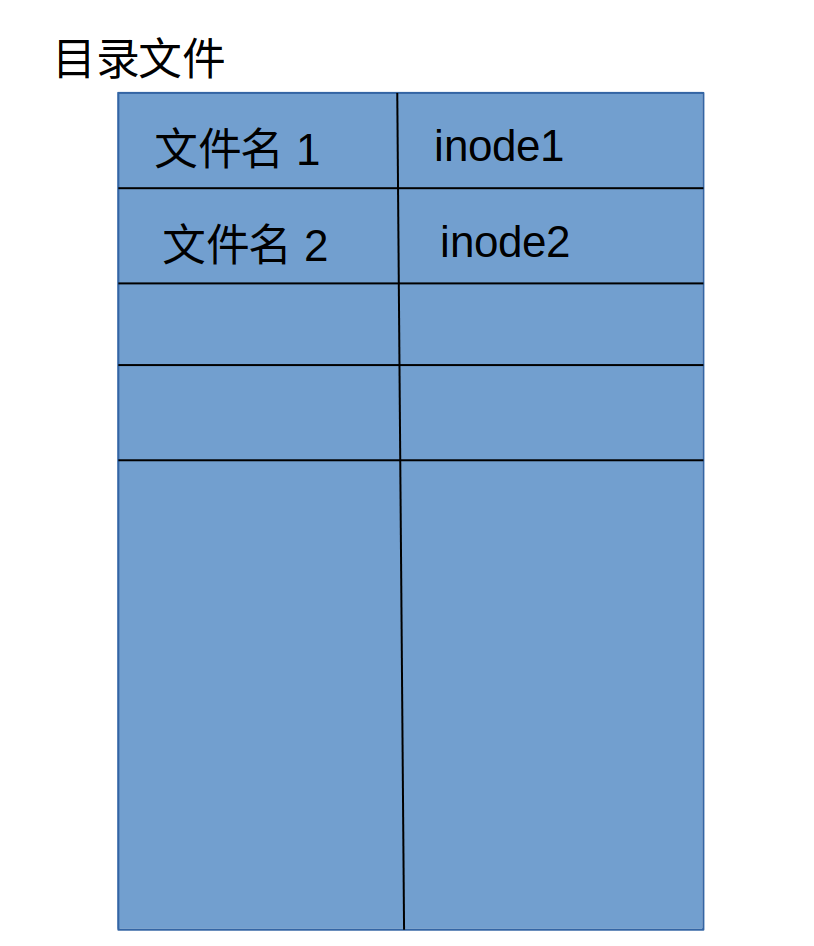
文件系统：文件的存储格式，文件的存储和管理方式。

FAT 微软闭源（静态单链表）

UFS UNIX开源（树）



目录文件 ：inode中并不包含文件名，文件名在目录文件中，目录文件也是普通的文件。(第二个inode块为根目录文件)

一条称为一个目录项

6硬链接 符号链接

一个硬链接就是增加一个目录项名字不同inode相同

符号链接就是增加一个inode

int link(const char \*oldpath, const char \*newpath);

int unlink(const char \*pathname);unlink可以用来产生匿名文件

int remove(const char \*pathname);

int rename(const char \*oldpath, const char \*newpath);

硬链接和目录项是同义词，不能给目录和分区建立硬链接。

符号链接可以跨分区，可以给目录建立。

int utime(const char \*filename, const struct utimbuf \*times);

7目录的创建和删除

int mkdir(const char \*pathname, mode\_t mode);

int rmdir(const char \*pathname);

8更改线程当前的工作路径

int chdir(const char \*path);

int fchdir(int fd);

char \*getcwd(char \*buf, size\_t size);

char \*getwd(char \*buf);

char \*get\_current\_dir\_name(void);

9读取目录内容

用统配符的形式解析一个目录

int glob(const char \*pattern, int flags,//统配符样式 ag:/etc/a\*.conf

int (\*errfunc) (const char \*epath, int eerrno),//出错回调函数

glob\_t \*pglob);//结果输出结构体

void globfree(glob\_t \*pglob);

DIR \*opendir(const char \*name);//打开目录以流的形式

DIR \*fdopendir(int fd);

struct dirent \*readdir(DIR \*dirp);//读取目录

int closedir(DIR \*dirp);//关闭目录

void rewinddir(DIR \*dirp);

void seekdir(DIR \*dirp, long loc);

long telldir(DIR \*dirp);

note:

“ls - - all” 长格式的作用是便于理解和记忆

如何使用touch命令创建 “-a” 文件？

“touch - - -a”会创建一个 名为 “-a”的 文件 “- -” 表示当前参数结束

或者“touch ./-a” 加上文件路径也可创建成功。

二、 表示系统信息的文件

三、进程环境