Project 6 File System设计文档

中国科学院大学

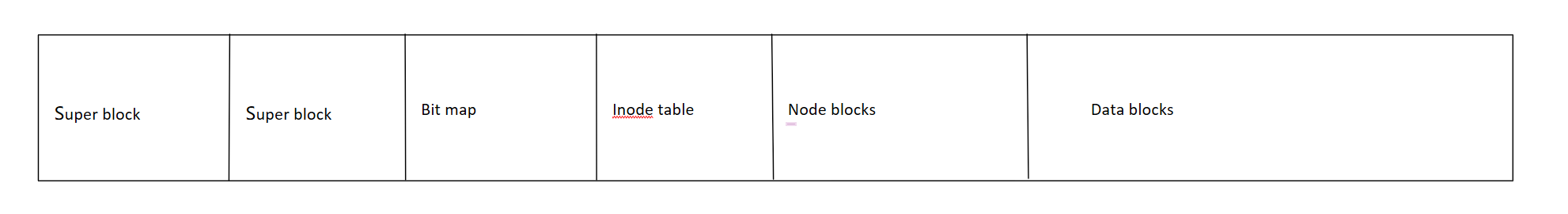
张旭

2018/1/21

# 文件系统初始化设计

请至少包含以下内容

1. **请用图表示你设计的文件系统对磁盘的布局**



一个inode结构实际大小284B，对齐196B，一个块有8个inode，inode占有15136个块。Inode块对应的map有15136B，每个字节对应一个块内的8个inode。

data有1025792个块，对应的map有128224B。

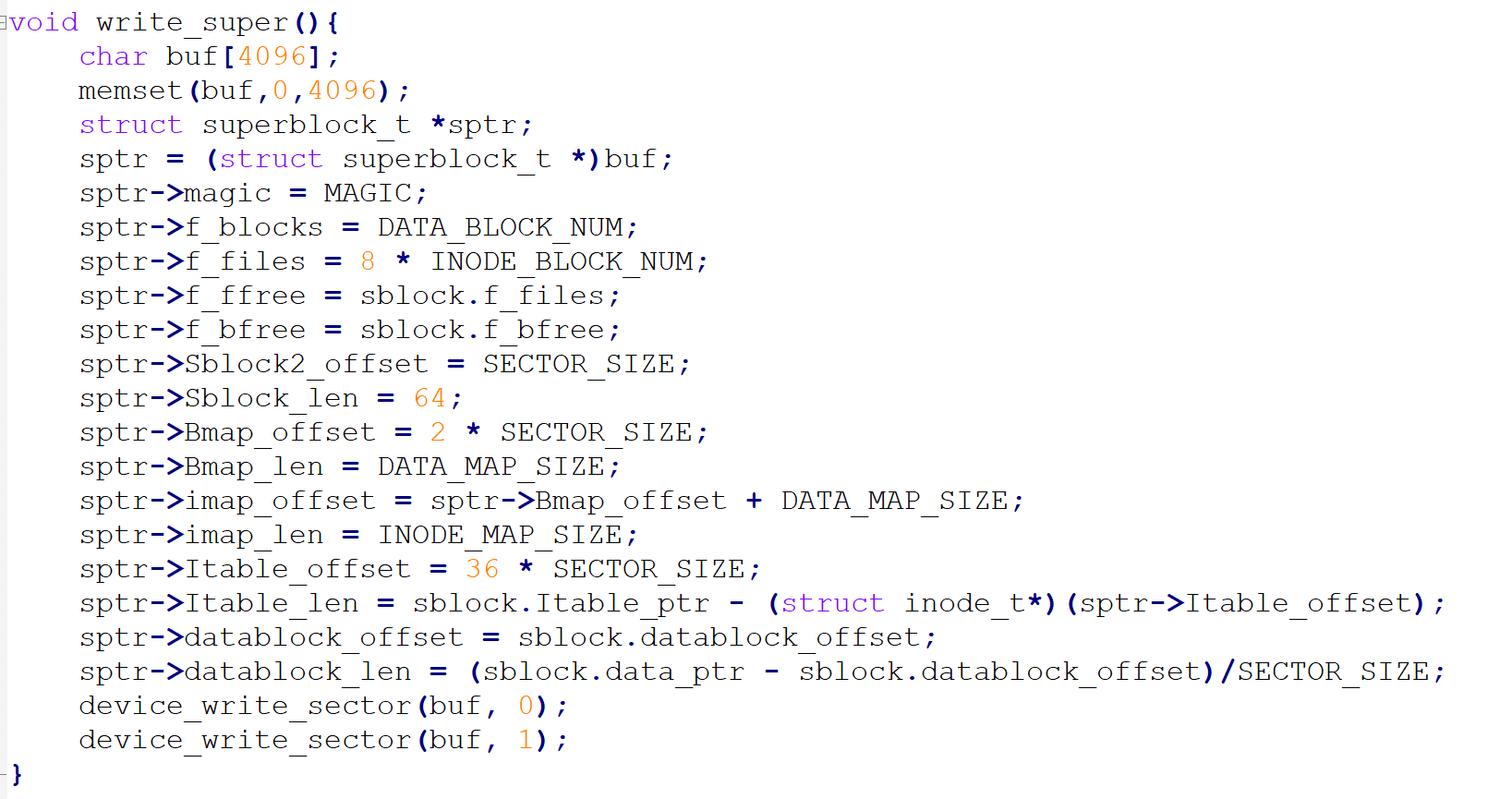
map有34块。

Dentry 结构实际大小268。一个块里面有15个dentry结构。

硬盘总共有1048576块，未利用7612块

1. **你如何实现superblock的备份？如何判断superblock损坏，以及当有一个superblock损坏时你的文件系统如何正常启动？**

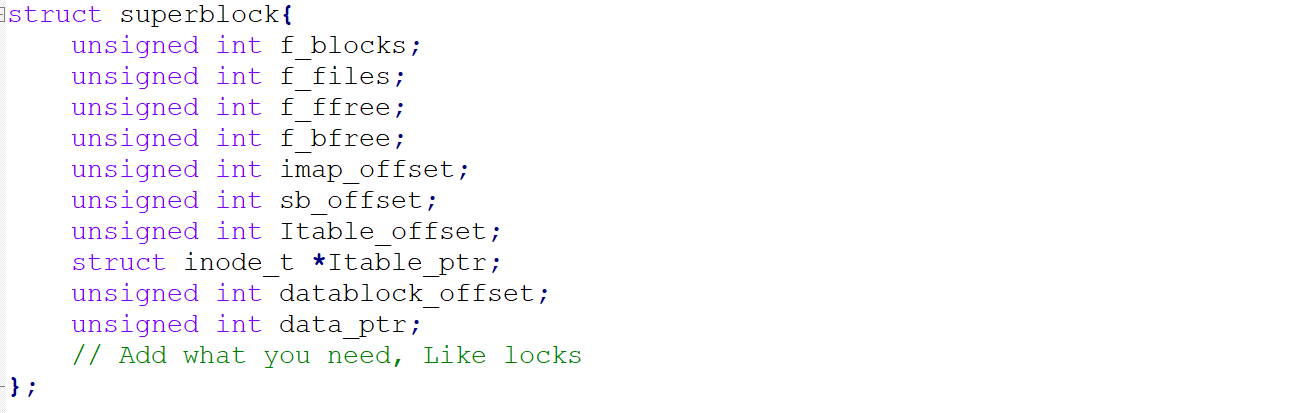
创建文件系统，以及每次在文件系统内创建文件时，我会备份superblock，备份函数如下：



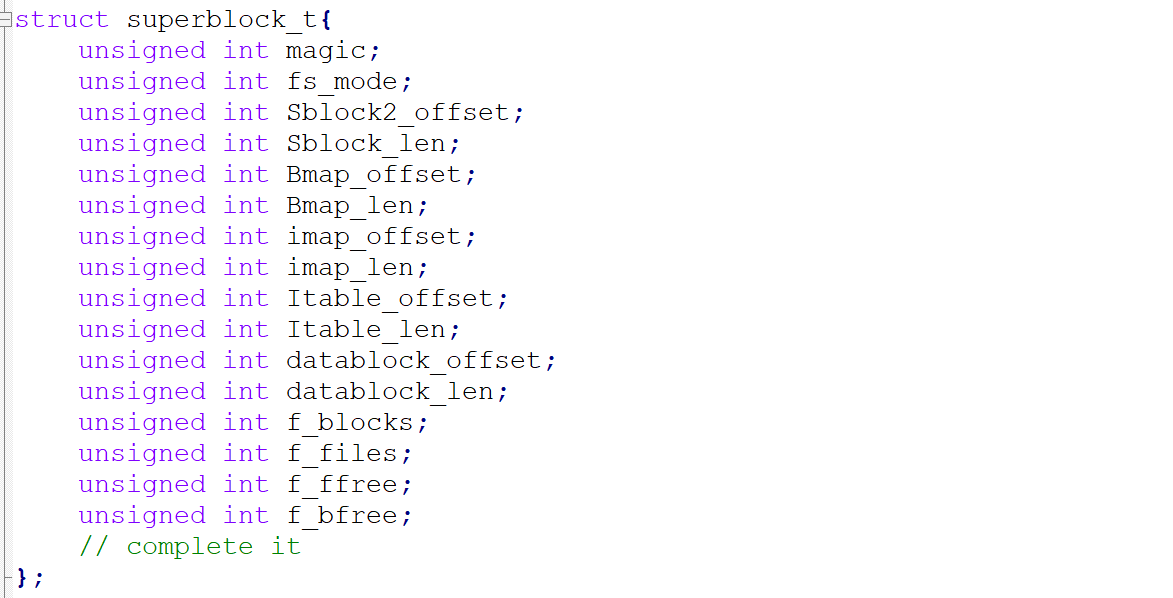
对于判断superblock损坏，我仅仅认为当magic不对时，该超块被破坏，实际上被破坏的情况有很多。当主超块被破坏时，我会读出备用超块进行初始化，同时修复主超块。

1. **请列出你设计的superblock和inode数据结构，并阐明各项含义。请说明你设计的文件系统能支持的最大文件大小，最多文件数目，以及单个目录下能支持的最多文件/子目录数目。**

内存中的超块结构如下：



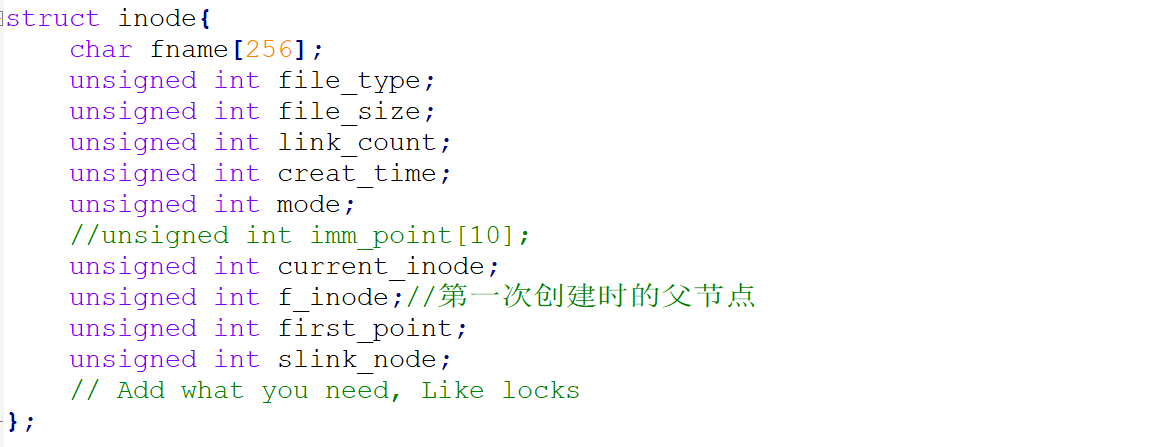
磁盘中的超块结构如下：



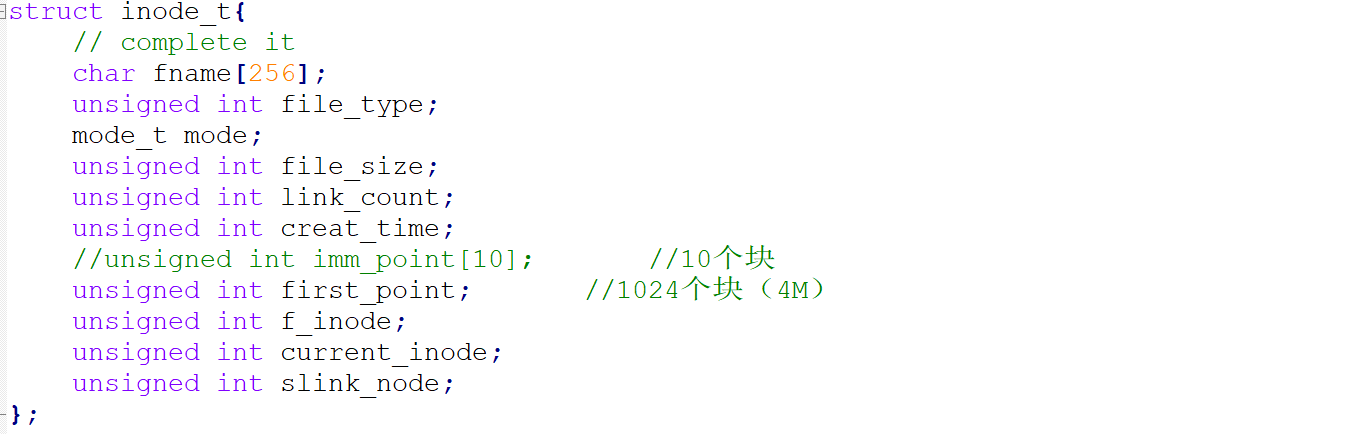
各项含义：

* f\_blocks：系统总共的块数
* f\_files：系统最大inode数
* f\_ffree：剩余可用inode数
* f\_bfree：剩余可分配的块数
* imap\_offset：inode对应的map偏移量
* imap\_len：inode对应的map大小
* Bmap\_offset：data blocks对应的map偏移量
* Bmap\_len：data blocks对应的map大小
* Fs\_mode：文件系统的权限
* Magic：文件系统的魔数
* Sblock\_len：超块的大小
* Sblock2\_offset：备用超块的起始地址
* datablock\_offset：数据块的起始地址
* datablock\_len：数据块的大小

内存中inode结构如下：



磁盘中inode结构如下：



各项的含义：

* fname：文件名称
* file\_type：文件的类型
* mode：文件权限
* file\_size：文件的大小
* link\_count：硬链接的个数
* create\_time：创建时间
* first\_point：指向数据块的一级指针
* f\_inode：父节点的inode编号
* current\_inode：当前节点的inode编号
* slink\_node：符号连接的inode编号

在我的设计中总共可以有121088个文件，每个文件通过一个一级指针指向它的数据，故一个文件最大4M。

一个目录下最多有1024个项。

1. **请说明你设计的文件系统的块分配策略，按需分配还是有设计其他分配策略？**

创建一个文件时，首先分配一个页面，由一级指针指向，然后再按需分配块。

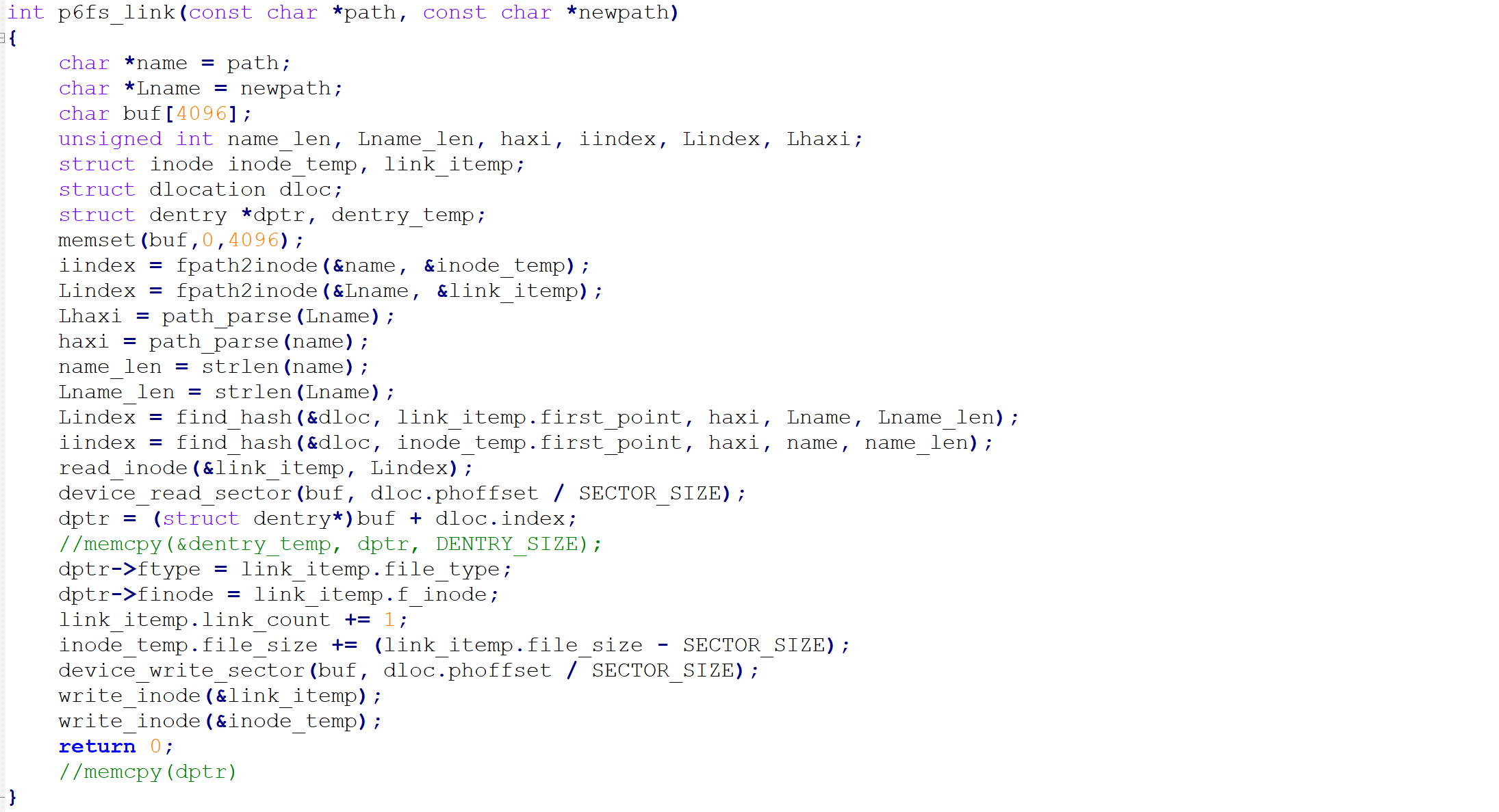
1. **设计或实现过程中遇到的问题和得到的经验（如果有的话可以写下来，不是必需项）**

由于对fuse一些结构和固有返回值不太熟悉，碰了很多坑，建议老师以后给与更多的信息。

# 文件操作设计

1. 请说明link和unlink的操作流程

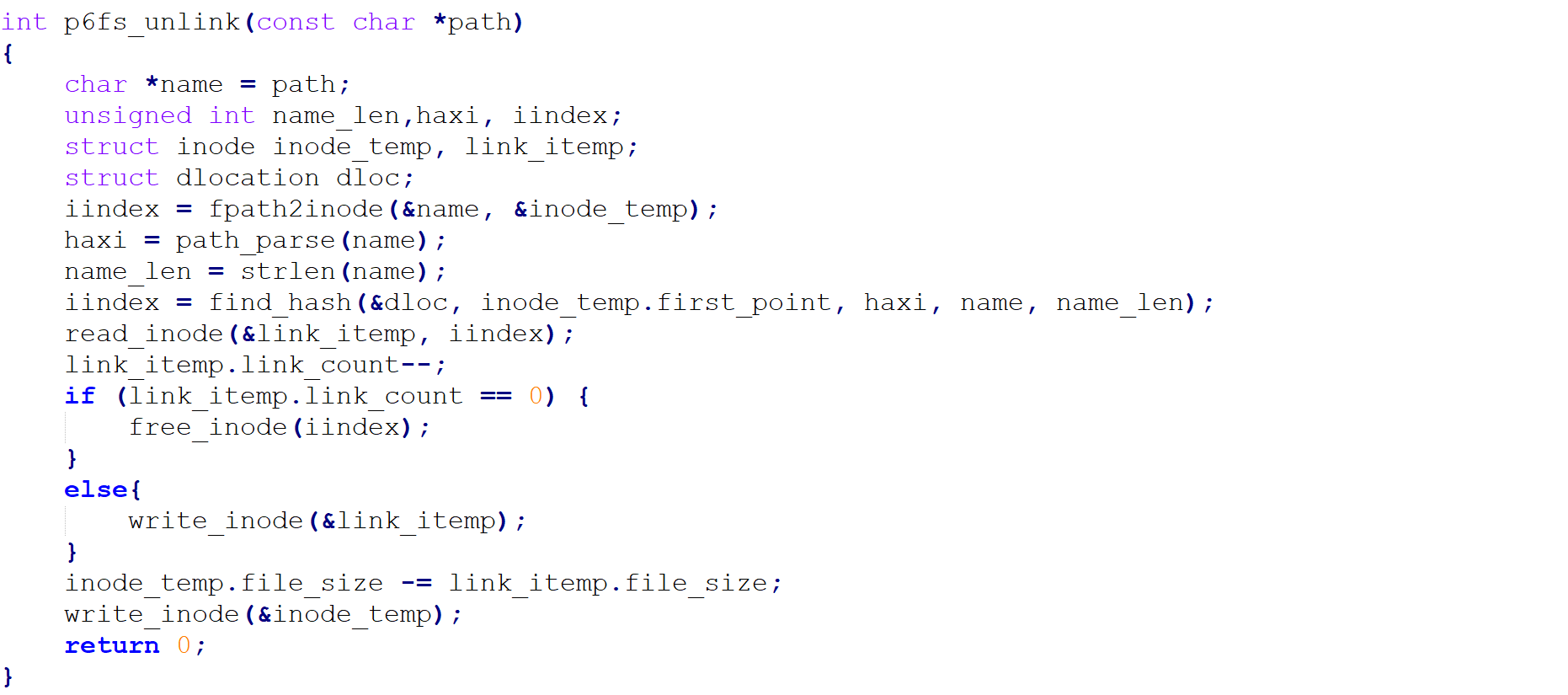
Link函数如下：



流程：

1. 找到两个路径对应文件的inode以及父节点的inode
2. 修改path父节点的目录项，将path对应的inode编号替换为newpath的inode编号
3. Newpath文件的link\_count加一
4. 写回磁盘

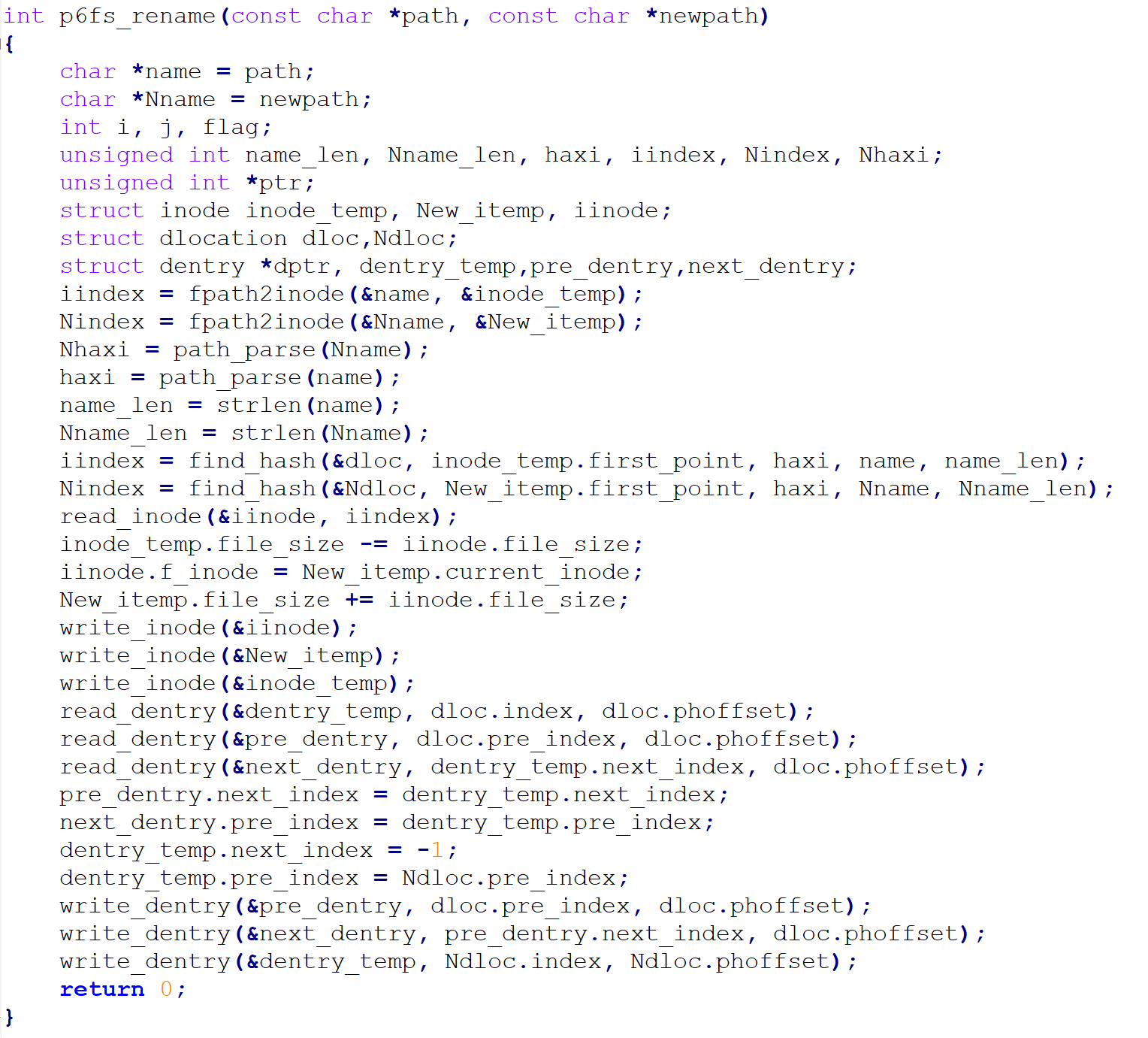
Unlink函数如下：



流程：

1. 找到path连接的inode
2. Link\_count减一
3. 若link\_count为0，则调用free\_inode函数，删除inode。
4. 改变Path的父节点的文件大小
5. 写回磁盘
6. 请说明rename涉及的操作流程

Rename函数如下：



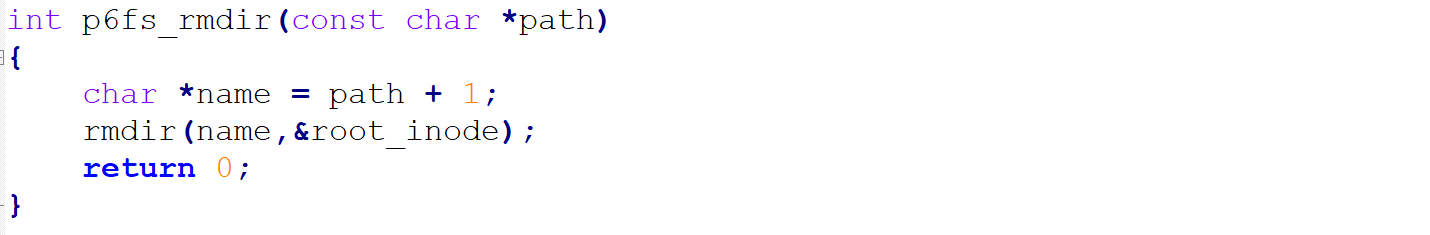
流程：

1. 找到两个路径对应文件的inode以及父节点的inode
2. 将newpath父节点的目录项根据path的inode进行修改（维护链表）
3. 清除path父节点的目录项
4. 写回磁盘

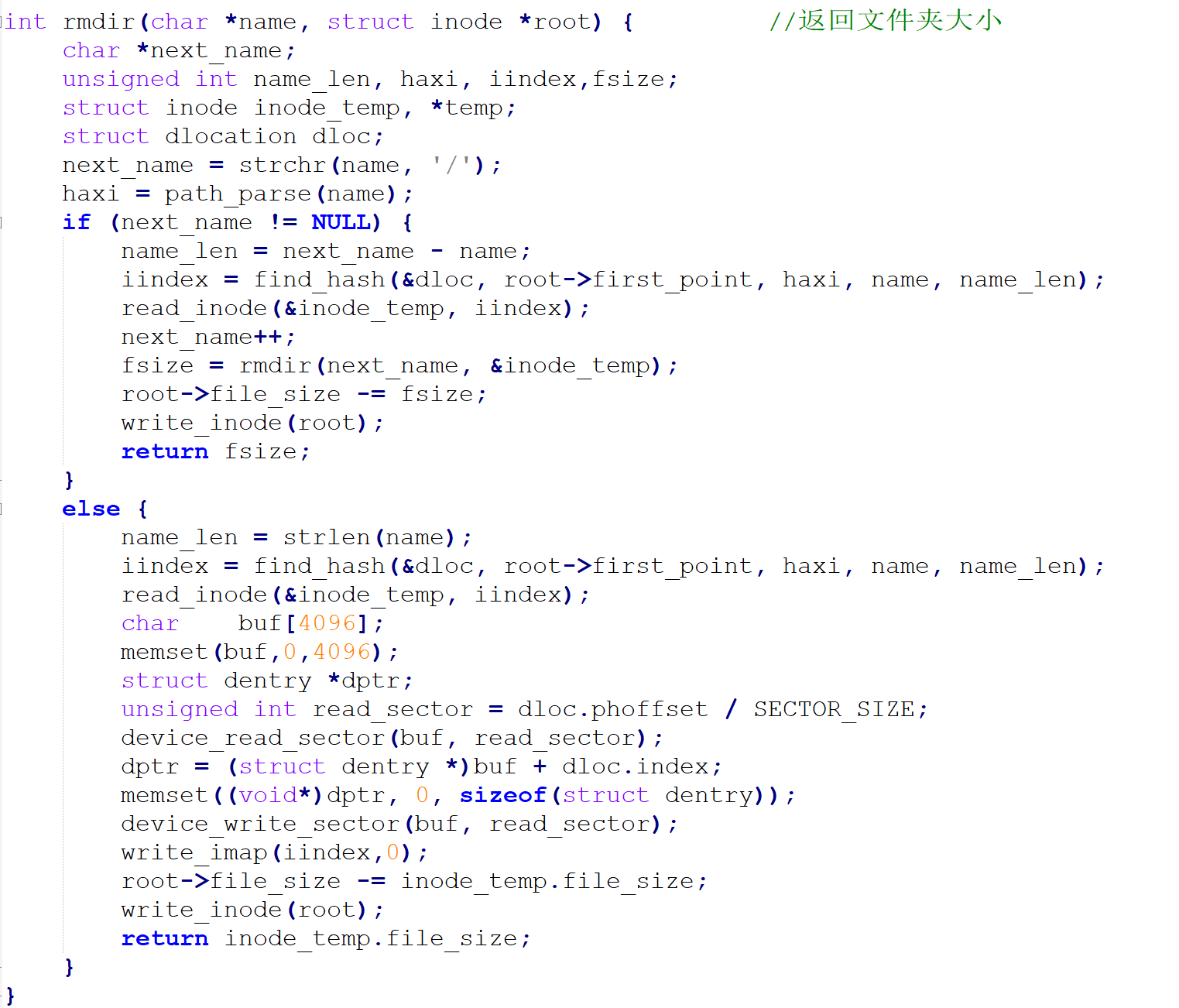
# 目录操作设计

1. 请说明rmdir的操作流程？

Rmdir函数如下：



我递归调用rmdir函数，以实现递归修改文件大小。函数截图如下：



对于递归的每一层，流程：

1. 找到对应的inode
2. 修改文件大小。并写回磁盘
3. 返回底层传来的文件大小

最底层：

1. 从父节点的目录中删除对应的目录项
2. 删除inode节点
3. 修改map，写回磁盘
4. 返回文件大小

# 关键函数功能

目录项布局设计：

一个目录一共63个组（630个块，总共1034个块），每组（由开头字母区分）可以有150个目录项，文件的命名允许52个字母（大小写），10个数字和‘\_’。

组内前126个目录项由第二个字符区分，后15个为溢出区

第三个字符决定放在高目录项或低目录项。

若前63个组内没查到，则到后63个备用组内查找（每组六个块，378个块），备用组有90个目录项，前63个目录项由第二个字符区分，后15个为溢出区。

dentey的fname[0]=‘\0’时，说明该目录项没有被使用。

所以本系统最多允许前两个字母相同的目录项54个。

Hash函数查找流程：

1. 若一级间指块未分配所在的地址，分配块，并返回。
2. 根据hash值找到链表起始位置，沿着链表往下查找，找到则返回地址。
3. 若未找到，在溢出区，备用区，备用溢出区依次寻找一个空位，维护链表并返回地址。

此外，为了方便，我完成了很多功能的函数封装，详情见代码。