****



**MIT xv6实验报告**

**——lab9 : file**

**学生姓名 胡轶然**

**学 号 3019244355**

1. **实验目的**
2. 实现二级间接块和符号链接。
3. 了解xv6系统中文件系统的工作原理。
4. **前期准备**
5. 切换git分支。
6. 阅读指导书第8章，了解与文件系统相关的知识。
7. 阅读相关代码。
8. **实验内容及实现步骤**

**任务1Large files**

1. **问题描述**

本任务要求将1个直接块替换为二级间接块，将单个文件的最大长度由(12+256)个块扩大为(12-1+256+256\*256)个块。

1. **思路与实现步骤**

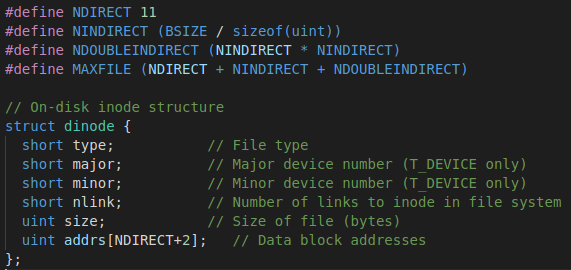
在Xv6的文件系统中，单个文件最多拥有12个直接块，1个间接块，每个直接块存储了1024B的数据，而间接块存储了1024/(sizeof uint)=256个块的指针。故单个文件的最大大小为(12+256)\*1024B。

要实现二级间接块，首先要修改fs.h文件中，与块数量、大小有关的宏定义，然后要修改fs.h、file.h文件中 inode、dinode结构体内部的文件块数组长度，最后还要修改fs.c文件中，读、写磁盘文件和释放磁盘文件空间的逻辑。

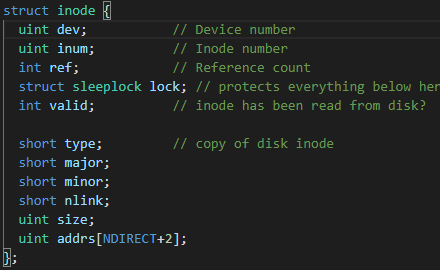
以下是具体实现步骤：

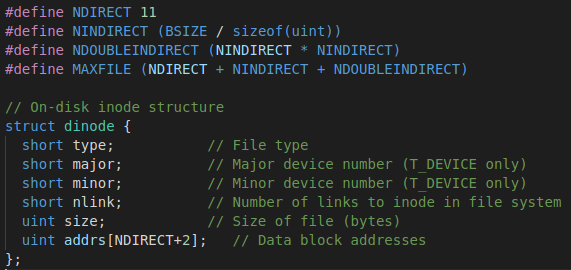
1. 修改fs.h文件中与块数量与大小相关的宏定义，减小直接块的数量。

具体包括：将直接块数量减1、定义二级间接块数量为256\*256、修改块总数量为11+256+256\*256。



1. 由于直接块数量改变，需要修改inode、dinode中对逻辑块数组的长度定义。其中dinode结构体存储磁盘中文件的信息，inode结构体通过valid位标记文件是否在内存中。



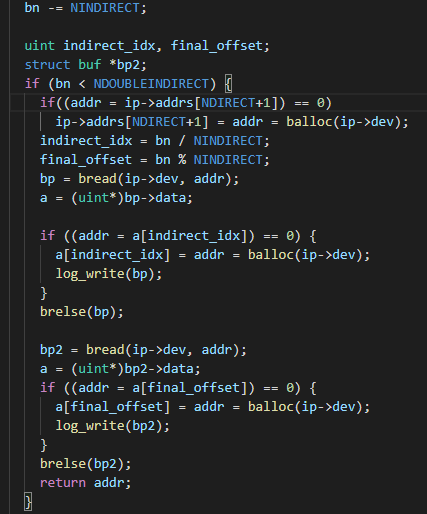


1. 然后在fs.c文件bmap函数中，仿照一级间接块的逻辑，添加读、写二级间接块的逻辑。

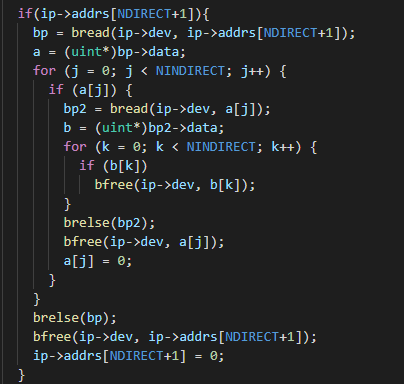
bmap函数在读写文件时被调用，其参数包括文件的指针和逻辑块号。读文件时，需调用bmap函数返回目标文件块在磁盘中的地址，写文件时，需调用bmap函数定位文件块，并通过调用log\_write将写操作提交到文件系统的log层。

以下是在bmap文件中添加的代码，具体修改内容包括：

1. 首先将逻辑块号调整为0~256\*256-1，这是为了便于计算各级文件块的数组下标。
2. 依次访问二级间接块、一级间接块和直接块在磁盘中的地址。
3. 若某个块的地址为空，则说明调用者要向磁盘中写入数据，此时调用balloc函数分配1个文件块，并调用log\_write将写操作提交到log层。
4. 若地址不为空，或刚为空的间接块分配磁盘空间，则要从磁盘中读取数据。若文件块是间接块，则将磁盘中的文件块以buf结构体形式读取到内存，并通过逻辑块号查找下一级文件块；若文件块是直接块，则调用者要读取文件，此时要返回文件块在磁盘中的地址。
5. 最后要释放内存中的buf结构体。



1. 然后要在itrunc函数中，添加释放二级间接块的逻辑，这部分代码与bmap函数读取文件相似，都是通过bread逐级读取间接块，并查找下一级文件块。我在此函数中通过bfree函数释放磁盘空间，并将间接块的指针改为”0”，防止出现野指针。



1. **问题与解决方法**

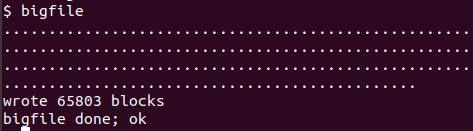
本任务较为简单，仿照原有代码实现即可。

**问题一**：如何继续扩大文件大小？

**解决方法**：

1. 实现更高级的间接文件块（牺牲速度）。
2. 增加高级间接块数量。
3. **增大文件块大小，对于高级间接块而言，可获得指数级的容量增长。**
4. **结果**

结果如下，共有65803个块，与预期相同。



**任务2：Symbolic links**

1. **问题描述**

本任务要求在xv6系统中实现文件的符号链接（软链接）。

1. **思路与实现步骤**

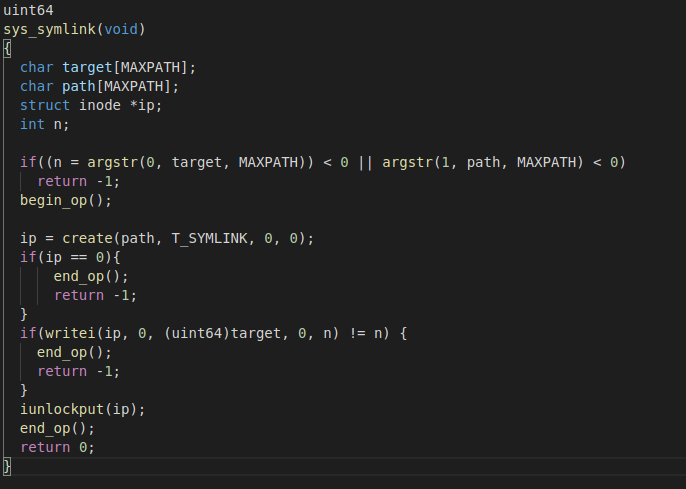
首先需修改[stat.h](https://github.com/PKUFlyingPig/MIT6.S081-2020fall/commit/c261d58dd5295aa1a90e7591327e1cdda02f1488" \l "diff-d4746551f09d954aa7b147464215020252e2d83aeeae698fec3c57c29f5eba57" \o "kernel/stat.h)和[fcntl.h](https://github.com/PKUFlyingPig/MIT6.S081-2020fall/commit/c261d58dd5295aa1a90e7591327e1cdda02f1488" \l "diff-d5d1199c439257624ccc6848ea975dff5bd28435c4799abccd554205058d04c0" \o "kernel/fcntl.h)文件，定义一种新的文件类型T\_SYMLINK以及对应的独热标识码O\_NOFOLLOW，然后要实现创建符号链接的系统调用函数sys\_symlink，和打开符号链接文件的系统调用函数sys\_open。

具体实现方法如下：

1. 添加系统调用，要修改的文件有:MAKEFILE、user.h、usys.pl、syscall.h，修改方式同Lab2。
2. 分别在[stat.h](https://github.com/PKUFlyingPig/MIT6.S081-2020fall/commit/c261d58dd5295aa1a90e7591327e1cdda02f1488" \l "diff-d4746551f09d954aa7b147464215020252e2d83aeeae698fec3c57c29f5eba57" \o "kernel/stat.h)和[fcntl.h](https://github.com/PKUFlyingPig/MIT6.S081-2020fall/commit/c261d58dd5295aa1a90e7591327e1cdda02f1488" \l "diff-d5d1199c439257624ccc6848ea975dff5bd28435c4799abccd554205058d04c0" \o "kernel/fcntl.h)文件中，添加文件类型T\_SYMLINK和独热标识码O\_NOFOLLOW的宏定义，其中，独热码用于指定链接文件的打开方式。在编写程序时，常量O\_NOFOLLOW的值有多种选择，只要不与其他文件类型重复即可，这里取0x010。
3. 实现sys\_symlink函数，该函数可根据传入的参数在指定路径下建立指定文件的符号链接。函数主要实现两个功能：创建符号链接文件、写入目标文件名。

具体操作为：

* 1. 首先使用createi函数在指定路径(path参数)创建符号链接文件（create函数会自动分配文件锁）。
  2. 然后将目标文件名(target参数)写入新创建的文件中。
  3. 最后释放文件锁。
  4. 若某步操作存在异常，就返回异常值”-1”。



1. 最后，还要在sys\_open函数中补充打开链接文件的逻辑，符号链接有两种模式：单次链接和多重链接，受O\_NOFOLLOW控制。若符号链接文件指向另一个符号链接，单次链接会直接将另一个符号链接当作正常文件打开，而多次链接会迭代地打开链接文件，直到遇到一个其它类型的文件，或迭代次数达到上限为止。函数的具体内容为：
   1. 读取链接文件内存储的目标文件名。
   2. 调用namei函数，根据文件名定位到目标文件。
   3. 若目标文件仍为链接文件，调用者打开了O\_NOFOLLOW标记，则支持多重链接，重复操作a~b。
   4. 若目标文件不是链接文件，或有O\_NOFOLLOW标记，则将目标文件信息存入变量”ip”中，释放文件锁并退出链接文件读取逻辑。
   5. 同时，还要避免出现循环链接的现象。因此，当检测到程序打开了一定数量的链接文件后（10次），就认定存在循环链接，此时将退出循环，释放锁并返回异常值”-1”。

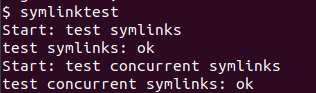


1. **问题与解决方法**

**问题一**：在create文件后为该文件**重复加锁**。

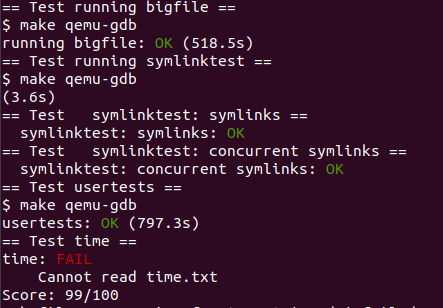
**解决方法**：阅读create函数后，发现**该函数在创建文件后会自动为创建文件的进程分配一个锁**，无需人为加锁。

1. **结果**



1. **实验结果**

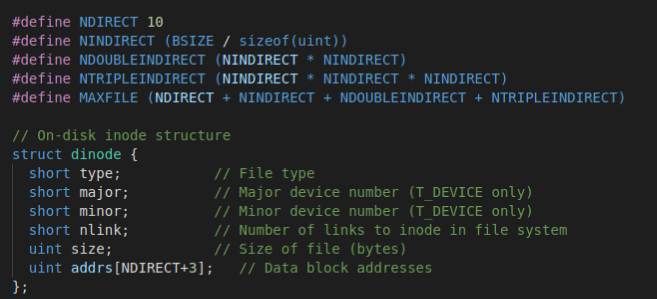
通过make grade测试。

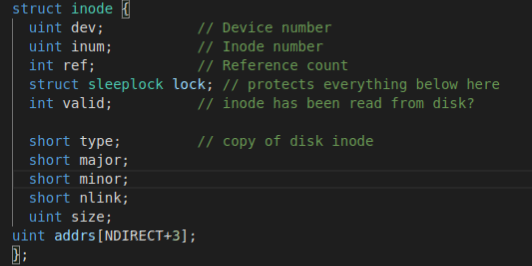


1. **Optional challenge exercises**
2. **思路**

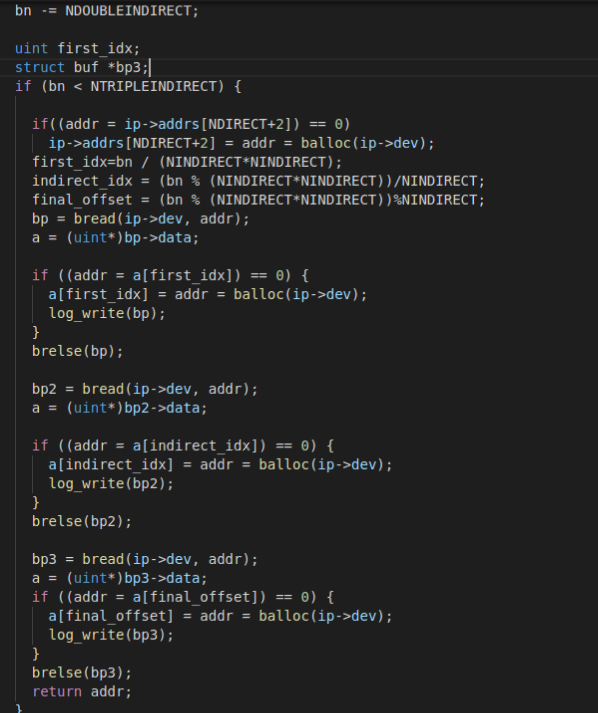
可选任务要求添加三级间接块，可仿照任务1，首先添加相关宏定义，然后修改两个结构体中的块数组定义，最后再fs.c文件中修改三级间接文件块分配、寻址和释放的逻辑。

1. **步骤**
2. 修改宏定义和文件结构体





1. 修改文件块的读写和释放逻辑

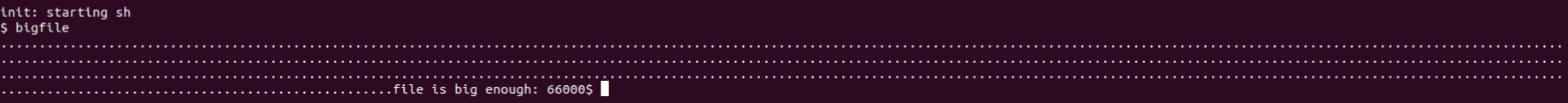




1. **结果**

由于三重间接块容量过大，难以通过既有方法测试，我修改了bigfile测试，使其在创建了一定数量的文件块后停止并输出信息。

**测试一**：创建66000个块



**测试二**：创建200000个块

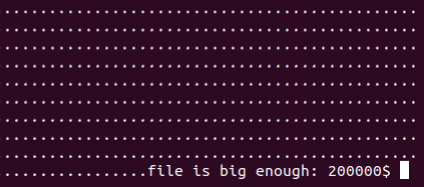
在创建了180000~190000个文件块后报错。

****

修改**param.h中的文件块数量**

****

**修改后创建成功。**

****

**测试三：创建1800000个文件块**

创建成功（约10小时）。

****