**实验报告**

**Lab 6**

姓名：施君豪

班级：20 人工智能

学号：20307140008

**实验报告填写要求**

1.请在每个exercise之后简要叙述实验原理，详细描述实验过程。可以使用中文进行描述，不对语言做要求。

2.请将你认为的关键步骤附上必要的截图。

3.有需要写代码的实验，必须配有代码、注释以及对代码功能的说明。

4.你还可以列举包括但不局限于以下方面:实验过程中碰到的问题你是如何解决的、实验之后你还留有哪些疑问和感想。

5.如果实验附有练习，请在每个练习之后作答，这是实验报告评分的重要部分。

6.Challenge为加分选作题。每个lab可能有多个challenge,我们会根据完成情况以及难度适当加分，具体情况会在课上说明。这部分的实验过程描述应该比exercise更加详细。

7.切勿抄袭亦或是去互联网复制粘贴答案。

【练习题模板】

1. Question
2. Code
3. Screenshot
4. Difficulties and solutions

实验练习一：**Large files**

**In this assignment you'll increase the maximum size of an xv6 file. Currently xv6 files are limited to 268 blocks, or 268\*BSIZE bytes (BSIZE is 1024 in xv6). This limit comes from the fact that an xv6 inode contains 12 "direct" block numbers and one "singly-indirect" block number, which refers to a block that holds up to 256 more block numbers, for a total of 12+256=268 blocks.**

**You'll change the xv6 file system code to support a "doubly-indirect" block in each inode, containing 256 addresses of singly-indirect blocks, each of which can contain up to 256 addresses of data blocks. The result will be that a file will be able to consist of up to 65803 blocks, or 256\*256+256+11 blocks (11 instead of 12, because we will sacrifice one of the direct block numbers for the double-indirect block).**

1. 思路与代码：

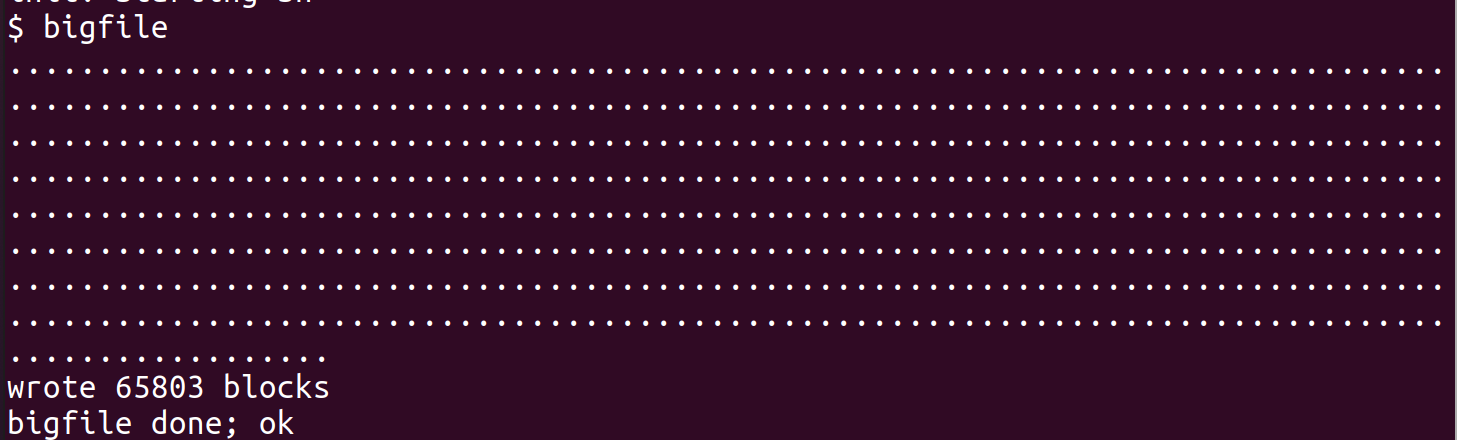
首先按照题目要求，将NDIRECT的值从12修改成11，并根据提示2修改fs.h与file.h中inode的结构。随后按照题目要求，修改maxfile的定义值，添加DOUBLEDIRECT的定义。

随后修改fs.c中的bmap() 函数，添加对第 NDIRECT 即 13 个块的二级间接索引的处理代码。处理的方法与处理第 NDIRECT 个块号即一级间接块号的方法是类似的, 只是需要索引两次。

最后修改itrunc() 函数，增加二级结构的释放。释放的方式同一级间接块号的结构, 只需要两重循环去分别遍历二级间接块以及其中的一级间接块。

源码已附附件中。

1. 运行截图



## 实验练习二：Symbolic links

**In this exercise you will add symbolic links to xv6. Symbolic links (or soft links) refer to a linked file by pathname; when a symbolic link is opened, the kernel follows the link to the referred file.**

**You will implement the symlink(char \*target, char \*path) system call, which creates a new symbolic link at path that refers to file named by target. To test, add symlinktest to the Makefile and run it.**

一、思路及代码：

首先按照hint1中的提示，在kernel/syscall.h, kernel/syscall.c, user/usys.pl 和 user/user.h.添加有关symlink系统调用的定义声明。在 kernel/stat.h中添加新的文件类型 T\_SYMLINK ，在kernel/fcntl.h 中添加新的文件标志位 O\_NOFOLLOW 。

然后实现 sys\_symlink() 函数, 首先通过 create() 创建符号链接路径对应的 inode 结构，同时使用 T\_SYMLINK 与普通的文件进行区分。然后再通过 writei() 将链接的目标文件的路径写入 inode中。

然后按照hint2的第二条修改sys\_open函数，由于对于符号链接一般情况下需要打开的是其链接的目标文件, 因此需要对符号链接文件进行额外处理，在原有代码之外follow\_symlink() 用来寻找符号链接的目标文件，功能为若需要寻找的是文件，则递归地去跟踪目标链接直至得到真正的文件。

除此以外，需要解决hint3中提出的问题:

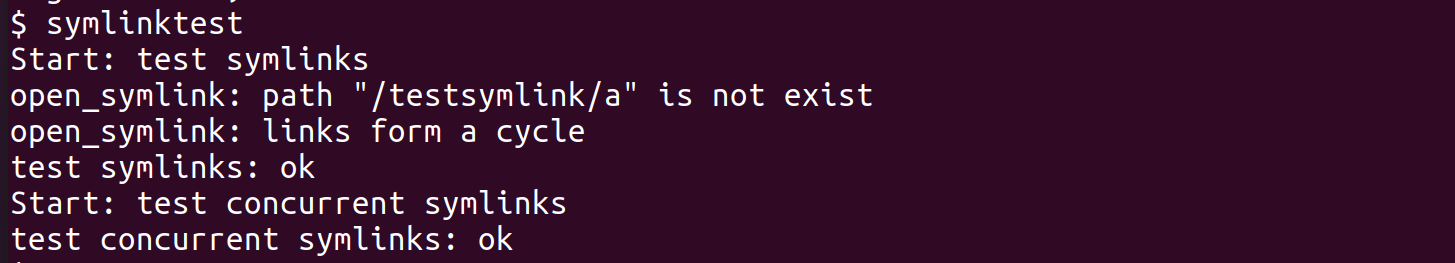
对于符号链接可能成环, 会一直递归地跟踪下去, 设置一个数组用于记录到过的节点，若新访问节点与中间内容重复，则判为循环， 另一方面是需要对链接的深度进行限制, 设置最大深度为10，超出则返回0..

对于禁止别的调用访问，则使用前面定义的NOFOLLOW位，若不满足要求则不执行上述操作

（if(ip->type == T\_SYMLINK && (omode & O\_NOFOLLOW) == 0) ）

源码已附在附件中。

1. 屏幕截图



1. 困难与解决办法

一开始对于标志位的运用不太熟悉，后来在查阅别的代码进行借鉴之后顺利解决。

**Part 3：The Ending**

通过本实验我了解了xv6系统对于文件系统的设置，同时对操作系统中内存的内容有了更深入的理解。文件系统一开始看起来还行，实际上手也有点磕磕碰碰。

以下为make-grade截图：

