思考题部分：

**思考题5.1 查阅资料，了解 Linux/Unix 的 /proc 文件系统是什么？有什么作用？ Windows 操作系统又是如何实现这些功能的？proc 文件系统这样的设计有什么好处和可以改进的地方？**

答：Linux系统上的/proc目录是一种伪文件系统（也即虚拟文件系统），存储的是当前内核运行状态的一系列特殊文件，用户可以通过这些文件查看有关系统硬件及当前正在运行进程的信息，甚至可以通过更改其中某些文件来改变内核的运行状态。

Windows运行 msinfo32命令可以查看硬件信息，通过任务管理器或者tasklist命令查看进程信息。

好处就是方便查看和管理内核的运行状态。需要改进的地方：写这些文件可以改变内核的状态，因而要慎重改动这些文件，其实用户是不能正确改变内核状态的，也不希望被用户改变到。

**思考题2**.2 **如果我们通过 kseg0 读写设备，我们对于设备的写入会缓存到 Cache 中。通过 kseg0 访问设备是一种错误的行为，在实际编写代码的时候这么做会引发 不可预知的问题。请你思考：这么做这会引起什么问题？对于不同种类的设备（如我们提到的串口设备和 IDE 磁盘）的操作会有差异吗？可以从缓存的性质和缓存刷新的策略来考虑。**

答：

写入缓存中，如果断电了缓存数据会丢失，缓存刷新会占用时间。

串口设备是通过数据线一位一位地传输，对数据线数量要求很低；IDE是并行接口设备，对于串行设备来说，缓存刷新速度慢，对于并行设备来说，并行刷新速度快。

**思考题3. 一个 Block 最多存储 1024 个指向其他磁盘块的指针，试计算，我们的文件系统支持的单个文件的最大大小为多大？**

答：1024\*4KB+1014\*1024\*4KB

**思考题4.查找代码中的相关定义，试回答一个磁盘块中最多能存储多少个文件控制块？一个目录最多能有多少个子文件？**

答：4KB/(NINDIRECT\*BY2BLK)= 16KB/(BY2BLK^2)=4个

**思考题5.请思考，在满足磁盘块缓存的设计的前提下，我们实验使用的内核支持的最大磁盘大小是多少？**

答：0xb0000000\*BY2BLK

**思考题6.阅读 user/file.c 中的众多代码，发现很多函数中都会将一个 struct Fd\* 型的指针转换为 struct Filefd \* 型的指针，请解释为什么这样的转换可行。**

答：存储结构和存储位数相同。

**思考题7.请解释 File, Fd, Filefd, Open 结构体及其各个域的作用。比如各个结构体会在哪些过程中被使用，是否对应磁盘上的物理实体还是单纯的内存数据等。说明形式自定，要求简洁明了，可大致勾勒出文件系统数据结构与物理实体的对应关系与设计框架。**

答：

**思考题8.阅读serve函数的代码，我们注意到函数中包含了一个死循环for (;;) {...}，为什么这段代码不会导致整个内核进入 panic 状态？**

答：

实验难点图示：

1. 磁盘驱动和用户态驱动程序。
2. 文件系统结构diskaddr: 计算磁盘块虚拟地址map\_block: 将指定的磁盘块映射到内存dir\_lookup: 查找某个目录下是否存在指定的文件
3. 文件系统用户接口，
4. 文件系统服务，文件 user/fsipc.c 中定义了请求文件系统时用到的 IPC 操作，user/file.c

文件中定义了用户程序读写、创建、删除和修改文件的接口

文件系统结构

Free\_block

Create file

diskaddr

map\_block:

dir\_lookup

STEP

04

文件系统服务

remove函数

STEP

03

文件系统用户接口

Open

read

STEP

02

STEP

01

读写系统调用，用户态磁盘读写驱动

sys\_write\_dev: 写外设

sys\_read\_dev: 读外设

ide\_write: 写磁盘

ide\_read: 读磁盘

体会与感想：

Lab5的文件系统涉及代码比较多，思路也比较复杂。第一次实验还比较简单，写了前两个练习40分去课上，也比较顺利地通过课上实验。当时比较纠结的是模仿系统调用函数read\_sector对磁盘进行读写操作，直接在用户态对物理地址进行写入和读取，然后需要处理一下写入和读取的字节数，这也是比较坑的地方。

Lab5-2的代码非常多，测试的两个进程调用后，各种初始化函数，各种写入读入函数也特别多，还不是很理解。Serve进程开启可以顺利执行，但是fstest进程没有顺利执行，有很多玄学错误，各种panic，地址异常。