第四次实例分析主要关注操作系统中文件系统的实现，xv6 的文件系统相关的系统调用有以下这些：

open read write close mkdir chdir mknod pipe dup link unlink fstat

这些系统调用的实现都在 sysfile.c 文件中，实际操作的代码主要在 file.c 和 fs.c 中，请大家将整个文件系统的操作都看完，再着重准备要讲的部分。

Linux 部分的内核版本和文件系统版本不做限制，建议看 Linux 2.6 以后的版本。

**第一部分 open 和 close：文件的打开（创建）和关闭**

入口代码：sysfile.c 中的 sys\_open 和 sys\_close 函数

关注问题：

1. open 和 close 系统调用的作用是什么，系统调用的参数有哪些，各自代表什么含义？

打开和关闭文件，在全局ftable中记录并在进程的表中分配一个文件描述符；open得到的inode最终要加锁。

char \*path; // 文件路径

int fd, omode; // 描述符，返回值；打开的模式

struct file \*f; // 全局表中存放的文件结构体

struct inode \*ip; // 创建的或读出的i-node的指针

1. open 和 close 中各种失败情况的原因都是什么

【open】创建：inode分配失败；其他：读取时namei未找到路径对应的inode；打开的是目录但是并非是只读操作；分配file结构体或文件描述符失败。

【close】描述符非法

1. sys\_open 函数中 f 变量是什么类型的变量，里面的各个成员变量都代表什么，在 sys\_open 函数中被设成了什么？和 fd 变量含义上有什么区别？

一个file结构体

成员：类型（inode或管道），引用次数，可读，可写，包装的inode或管道，I/O偏移量

type：设为inode；ip：设为传入的inode指针；偏移量：设为0；读写位：按传入的模式MASK设置

一个file可以在同一table或不同进程的多个table中多次出现；fd是描述符，是表中的一个下标

1. sys\_close 函数中 proc->ofile[fd] = 0 这一句代表了什么？ofile 变量还有什么时候被赋值过？

把描述符置为无效，释放描述符

fdalloc的时候赋值

1. filealloc 函数的作用是什么，ftable 是一个什么数据结构，里面总共多少项？

filealloc在全局ftable中记录打开文件

ftable中包含一个file数组和一个自旋锁

一共NFILE=100项

1. fdalloc 函数的作用是什么？

为给定的文件分配文件描述符；之后调用者对文件的访问都通过它来进行

1. （进阶题）请比较 linux 内核代码中的 open 和 close 函数，它们的实现和 xv6 中有哪些不同，为什么？（包括 open 和 close 进一步调用的函数，以及类似 ftable 的数据结构）

**第二部分 read 和 write（part 1）：read 的实现和 buffer cache 管理**

入口代码：sysfile.c 中的 sys\_read 和 sys\_write 函数，file.c 中的 fileread 和 filewrite 函数，fs.c 中的 readi 和 writei 函数

关注问题：

1. fileread 和 filewrite 的三个参数分别是什么？

文件指针、读取/写入的目标地址、字节数

1. fileread 和 filewrite 函数中对 f->type 的判断代表什么？（f->type == FD\_PIPE 的情况由负责第五部分的同学来讲）文件的权限管理是怎么样实现的？

判断文件是管道还是inode，从而判断是从缓冲区copy（向缓冲区write）还是做磁盘操作

open操作时已经设定了readable和writable属性，检查即可

1. f->off 变量代表了什么？它的值是怎么变化的？

打开文件的偏移量

初始值为0，读写过程中维护（f->off += r）

1. readi 和 writei 函数的参数 ip，它的变量类型是什么，包含的成员变量都是什么含义？

inode结构体的pointer

uint dev; // 设备号

uint inum; // ino

int ref; // 引用次数

struct sleeplock lock; // 睡眠锁，保护下面的成员

int valid; // inode已经从磁盘读出

short type; // inode类型（目录/文件/设备）

short major;

short minor; // 主从设备号

short nlink; // inode的硬链接次数

uint size; // 文件大小

uint addrs[NDIRECT+1]; // 直接块指针

1. 请逐行介绍 readi 函数中文件操作实际执行到的内容。（writei 与 readi 功能相似，请也简要对比介绍一下，详细介绍由第三部分同学完成）

对于设备：检查major号/可读属性，然后调用设备的读函数

对文件（目录）：检查offset和size的相对关系，以及尾地址的合法性；每次读一块，先从块号找到目标块的地址然后拷贝；如果是读操作，把这个块放到MRU位置；如果是写操作，还要记录写日志

1. bread 函数最终调用 bget 函数，请介绍 bget 函数的功能。特别是 bget 函数中设计 buffer cache 的替换，请点明替换算法是什么。

bget在块缓冲区中找到缓冲块，如果此缓冲块已经有进程占用，则睡眠当前进程，等待唤醒。如果bget没有找到相应的块缓冲结构，则在缓冲区中找到一个无效的块缓冲区并返回，由bread调用iderw来将数据读入。如果缓冲区满，bget简单地panic。

替换算法是LRU。

1. buffer cache 中的 dirty 块是由哪个函数写入磁盘的？（跟踪 B\_DIRTY，具体内容由第三部分介绍）

iderw函数

1. （进阶题）linux 内核代码中 buffer cache 的替换是如何实现的？替换算

法和 xv6 有何不同？

**第三部分 read 和 write（part 2）：write 的实现和日志管理入口代码：fs.c 中的 writei 函数**

关注问题：

1. writei 函数中调用了 log\_write 函数，它的功能是什么？

在内存中记录块的扇区号；标记buffer的B\_DIRTY位；数据并不会立即写入磁盘，只有当调用commit来提交日志时，磁盘操作才会正式开始磁盘操作

同一次事务中对一个块的多次写被合并到一个日志项中（absorption）

1. 阅读 end\_op 函数代码，介绍 end\_op 函数何时会被调用，end\_op 函数的

功能是什么。

outstanding次数减1，如果是最后一个outstanding（count为0），即没有任何进程正在操作log时，调用commit提交日志。

1. end\_op 函数调用了 commit 函数，commit 函数的流程是什么，每个语句

分别完成了什么功能？（介绍相关调用函数）

四个阶段

write\_log()：把修改后的块从缓存写入磁盘的日志

write\_head()：真正提交的点，把header块写入磁盘

install\_trans()：从日志中读出每个块然后写到文件系统中的相应位置

write\_head()：清掉log header

1. 阅 读 initlog 函 数 ， log 的 初 始 化 是 何 时 完 成 的 ， 里 面 调 用 了recover\_from\_log 函数，为什么要调用它，它的作用是什么？

初始化在启动时，运行第一个用户进程前调用

从当机中恢复；检查是否已经commit，然后完成end\_op的后两个操作

1. （进阶题）linux 代码中实现了哪些日志文件系统，和 xv6 中的日志管理

有什么不同？

**第四部分 mkdir，chdir 和 mknod：文件目录的操作和设备文件的创建**

入口代码：sysfile.c 中的 sys\_mkdir 函数，sys\_chdir 函数以及 sys\_mknod 函数

关注问题：

1. 请逐行介绍 sys\_mkdir, sys\_chdir 和 sys\_mknod 函数的功能。

sys\_mkdir：创建一个目录，类似open里的create

sys\_chdir：drop掉当前所在目录的inode cache，然后更改当前目录working directory的inode指针（curproc->cwd）

sys\_mknod：创建一个特别设备文件，major和minor号由参数给出

1. create 函数调用了 dirlookup 函数，它实现了什么功能，是怎么实现的？

在目录中查找dentry；找到的话把poff设为这个entry的byte offset

在directory这个块里面逐项读取

1. create 函数还调用了 ialloc 函数，它的功能是什么？

在给定的设备上分配一个inode，类型标记好，日志里面记上allocated，用brelse标记为referenced

1. namei 函数和 create 函数最终都调用 fs.c 文件中的 namex 函数，请逐行介绍它的功能。

按路径查找对应的inode

判断是绝对路径还是相对路径，然后一级级往下走

1. （进阶题）linux 内核代码中，和 dirlookup 函数功能类似的函数是什么，实现上有何不同？
2. （进阶题）linux 内核代码中，和文件目录相关的系统调用还有什么，它们

的实现是怎么样的？

**第五部分 pipe：管道的实现**

入口代码：sysfile.c 文件中的 sys\_pipe 函数；pipe.c 中的 pipealloc、pipeclose、 pipewrite、piperead 函数

关注问题：

1. pipe 类型的文件操作实现的是什么样的功能？pipealloc 函数的两个参数分别代表什么？

管道：一对文件描述符，在一端写数据，另一端可以从缓冲区读出来（进程间通信机制）

读方和写方文件的指针

1. piperead和pipewrite函数的流程是什么样的？请介绍出报错（return -1）的原因，以及核心语句。

pipewrite：获取锁，然后写，如果满了就唤醒读者； 如果没人读或者进程被杀掉了就释放锁返回-1

piperead：申请锁，获取到锁之后发现nread不等于nwrite然后就读；如果进程被杀掉了就释放锁返回-1

本质上是个管程做的生产者消费者问题

1. pipe 类型的文件读写和 inode 类型的读写有什么区别？pipe 类型操作的实现目的是什么，有什么优点？

通过piperead和pipewrite而非inode实现，不需要磁盘操作，只是缓冲区读写

进程间通信

1. （进阶题）linux 内核代码中 pipe 的实现和 xv6 有哪些不同？

**第六部分 剩下的一些文件操作（dup，link，unlink 和 fstat）**

！特别注意：这一部分 xv6 的代码已经比较少了，请同学尽量完成一下进阶题的内容。特别是上课有介绍过，但是 xv6 代码中没有涉及的部分

入口代码：sysfile.c 文件中的 sys\_dup, sys\_link, sys\_unlink 以及 sys\_fstat 函数

关注问题：

1. dup 系统调用的功能是什么，为什么要设计这样一个系统调用？

复制一个文件（duplicate）；给他一个新的文件描述符，并给全局表中记录的文件引用次数+1

可以给写操作重定向到另一个描述符

1. link 和 unlink 两个系统调用实现了什么功能？请介绍它们的实现流程。

硬链接、链接次数-1

1. fstat 系统调用返回了文件的哪些状态？

short type; // Type of file

int dev; // File system's disk device

uint ino; // Inode number

short nlink; // Number of links to file

uint size; // Size of file in bytes

1. （进阶题）linux 内核代码中有哪些和文件系统相关的系统调用，除了第三部分同学介绍的那些之外，比 xv6 中还多出了哪些，请一一介绍它们的功能。（特别注意 fsync 的实现）