***JOS Lab5 文档***

杨健邦 515030910223

*本文档描述了本次LAB各个Exercise实现的方法或者我自己的理解。至于对Question的解答以及以及Challenge的实现方法，请阅读answers-lab5.txt。*

**Disk Access**

**Exercise 1**

在env\_create中判断type是否等于ENV\_TYPE\_FS，如果等于的话，就将e->env\_tf.tf\_eflags |= FL\_IOPL\_3，这表示的是在这个env中，RING 3及以上特权可以进行IO操作，以及用户态可以操作IO。

## The Block Cache

**Exercise 2.**

先用sys\_page\_alloc分配一块物理页，接着用ide\_read将文件的内容读入物理页，最后sys\_page\_map清除dirty\_bit。

## The Block Bitmap

**Exercise 3.**

从0开始一直到super->s\_nblocks，逐一搜索bitmap，直到找到第一个为0的位，将其置为1，并且立刻将bitmap block flush，然后再返回这个位在bitmap中的位置（第几个bit）。

## File Operations

**Exercise 4.**

对于file\_block\_walk，先判断filebno有没有超出NDIRECT以及NINDIRECT之和，如果是在NDIRECT之内的，直接返回f\_direct数组中的结果。如果在NINDIRECT里面，则根据参数，如果有需要的话分配一个新的物理页来存放NINDIRECT数组，并且返回数组中的结果。

对于file\_get\_block，其逻辑与file\_block\_walk类似，如果文件相对应的block不存在的话，还需要调用alloc\_block的方法分配一个新的block。

## Client/Server File System Access

**Exercise 5.**

在serve\_read中， 先根据文件id找到相应的openfile对象，通过file\_read的方法读出文件的内容，并修改fd中的offset，要注意读取字符的数量等于MIN(req->req\_n, PGSIZE)，防止buf溢出。

在devfile\_read中， 将要读的文件id以及要读的字符数设置好，并将IPC的方法标识号设置为FSREQ\_READ，如果IPC的返回值大于等于0，则将返回的内容move到buf中。

**Exercise 6.**

在serve\_write中，先根据文件id找到相应的openfile对象，通过file\_write将内容写入文件，并修改fd中的offset，要注意写入的字符数量等于MIN(req->req\_n, PGSIZE - (sizeof(int) + sizeof(size\_t)))，防止buf溢出。

在devfile\_write中，将要写的文件id以及要写的字符数设置好，并将buf的内容move到fsipcbuf.write.req.buf中，move的字符数量为MIN(n, PGSIZE - (sizeof(int) + sizeof(size\_t)))，防止buf溢出，最后将IPC的方法标识号设置为FSREQ\_WRITE，调用IPC。

## Client-Side File Operations

**Exercise 7.**

先调用fd\_alloc得到一个未被使用的fd，这里不需要为这个fd分配一块物理页。然后调用IPC，方法标识号为FSREQ\_OPEN，这个IPC会自动为这个刚分配的fd分配一块物理页，并将相应文件打开，绑定到这个fd上。最后通过fd2num(fd\_store)返回这个fd的index。要注意IPC返回失败结果时，要关闭fd，释放掉IPC为这个fd分配的物理页。

## Spawning Processes

**Exercise 8.**

在sys\_env\_set\_trapframe中，先检查tf的地址是否有效，然后将tf拷贝到对应的env的tf中，最后将env的tf的cs或上0x3、tf\_flags或上FL\_IF，以确保新的env是运行在用户态并且中断是开启的。