Lab5 Solution

516030910006 方俊杰

Exercise 1.

init.c 需要创建 fs env，给定其 type 为 ENV_TYPE_FS，修改 env_create 把 fs env 赋予 IO 特权。在判断为对应类型的时候 修改 env_alloc 创建的页面

```
env->env_tf.tf_eflags |= FL_IOPL_MASK
```

Question

1. 不需要。在不同的用户态环境切换的时候会把包括IO权限位所在的 eflags 寄存器等寄存器作为 trapframe 保存并在下一次调度到时恢复，不需要额外操作就能保证其不被改变

Exercise 2.

实现 fs_bc.c 中的 bc_pgfault & flush_block 函数。

- 对于前者，相当于需要去 disk 读取数据放到 block cache。先用 sys_page_alloc 在 fs env 的页表分配页，然后用 ide_read 把 disk 读入到对应得虚拟地址。
- 对于后者先用写好的 接口检查 是否 map 和 是否 dirty，用 ide_write 写回到 disk 然后用 sys_page_map 修改 perm

Exercise 3.

以 free_block 为模板实现 alloc_block，在 bitmap 找到一个 free disk block ，返回其 块号。分配的时候要把修改后的 bitmap 立刻 flush 回 disk 保证一致性。bitmap 中 1 的 bit 是 free。找到后 bitmap[i/32] &= ~(1<<(i%32)) 设置为被使用

Exercise 4.

实现 file_block_walk & file_get_block。

- 前者是 用 filebno （文件 f 中的索引号找到对应的 disk block number），如果在 indirect 的块，就要判断 alloc 来选择是否创建。最后都是把存有对应的 disk block number 的地址写到 *ppdiskbno 上，这样在 file_get_block 里就可以为 真正存数据的 block 分配块，并能够修改 struct File 或者 indirect block 中对应的指针。
- 后者是用 前者 找到对应 filebno 的指针，然后如果没分配，就创建并写回。然后把内容写到 *blk。

Exercise 5.

实现 serve_read 在 fs/serv.c
该函数的繁重工作已经由 file_read 完成，所以只需要提供用于文件读取的 RPC 接口。查看 serve_set_size 了解服务器功能构建。

查看 上述的 union Fsipc，可以看到是对不同的 IPC 号有不同的 struct 表示需要的参数。
查看 其他 server 端的函数，可以知道是对 file_* 函数的封装，先是 openfile_lookup 读取 openFile 结构，然后用 file_read 读取文件数据到 ret 的页就可以，关注调用所需的 参数就行了，offset 在 fd 里面，毕竟是 env 不共享的
然后更新对应的 offset

Exercise 6.

实现 fs/serv.c 的 serve_write & lib/file.c 的 devfile_write
模仿 serve_read 和 devfile_read 写即可
注释说要写的可能很大，但是允许写的总是固定，要检查，就是req_n 的大小要比 sizeof(req_buf) 小

Exercise 7.

spawn 依赖于新的 syscall 即 sys_env_set_trapframe 来初始化新创建的环境的状态。实现之，记得 dispatch
按照注释，获取到 env，然后设置

字段	修改
tf 为传入的参数	e->env_tf = *tf
CPL 为 3	e->env_tf.tf_cs |= 3
开启中断	e->env_tf.tf_eflags |= FL_IF
IOPL 为 0	e->env_tf.tf_eflags &= ~FL_IOPL_MASK

Exercise 8.

修改 duppage in lib/fork.c & 实现 copy_shared_pages in lib/spawn.c 适应上述共享。

- 前者：如果一个 page table entry 设置了 PTE_SHARE 位，则直接拷贝映射。（应该使用 PTE_SYSCALL，而不是 0xfff 来屏蔽页表项 的相关位，0xfff还会获取 accessd & dirty 位）。优先检查了 PTE_SHARE 位，成立则直接映射，用 PTE_SYSCALL来屏蔽作为 perm
- 后者：循环遍历当前进程中的所有页表条目（就像fork一样），将任何设置了PTE_SHARE位的页面映射复制到子进程中。检查 UTEXT 到 USTACKTOP 的每个页，PTE_SHARE 成立也 用 PTE_SYSCALL来屏蔽作为 perm

运行 make run-testpteshare 和 make run-testfdsharing 有对应输出

Exercise 9.

在 kern/trap.c，调用 kbd_intr 处理 trap IRQ_OFFSET+IRQ_KBD；调用 serial_intr 处理 trap IRQ_OFFSET+IRQ_SERIAL。两个函数已经在 console.c 写好了，但是 trap_dispatch 没有分发，分发之即可。

运行 make run-testkbd 类似复读机

Exercise 10.

shell 不支持 I/O 重定向. 如果能运行sh < script 而不需要所用命令都靠手动输入 会更好，在 user/sh.c中增加 I/O 重定向 <的功能。
原来 > 的重定向已经写好了，是重定向到 stdout 1
模仿着打开 t 的文件，然后重定向 dup 到 stdin 即 0 上就好了

Challenge

- 实现 FIFO 的 page eviction 策略
- 对于 block 0 1 用于 boot、super、bitmap 的 block 需要频繁访问，不进行淘汰；对于其他 block，将所有访问的记录下来，放入一个大小为 BCSIZE 的队列（实际实现时使用的是数组），当队列满时就淘汰队列头部 block 以放入新的 block。
- 实现细节：
    1. 使用条件编译，设置宏 EVICTFLAG 在 fs/fs.h，作为编译开关
    2. 当宏开启时，定义 BCSIZE 作为 bc 的大小，定义相应大小的数组来实现队列，额外定义 first_in 变量表示队列头部。
    3. 在开启淘汰策略的情况下，对于 block 0 1 用于 boot、super、bitmap 的 block 需要频繁访问，不进行淘汰。对于其他 block 放入队列中，当队列满时就淘汰队列头部 block（即 first_in 指向的块）。
    4. 对于要被淘汰的块，如果是 dirty，先 flush，然后 map 新地址防止引用计数清零被 free，再 unmap 被淘汰地址，然后从磁盘读入块到新地址，最后自己映射到自己清空 dirty 位。
- 开启宏，编译运行 make run-testfile 输出如下

```
FS is running
FS can do I/O
Device 1 presence: 1
permanent block 1
permanent block 1
permanent block 1
block cache is good
superblock is good
permanent block 2
bitmap is good
alloc_block is good
EVICT: No. map block 317 at 0x1013d000
EVICT: No. map block 318 at 0x1013e000
file_open is good
EVICT: No. map block 3 at 0x10003000
file_get_block is good
file_flush is good
file_truncate is good
file rewrite is good
serve_open is good
file_stat is good
file_read is good
file_close is good
stale fileid is good
EVICT: No. map block 320 at 0x10140000
file_write is good
file_read after file_write is good
```

```
open is good
EVICT: No. map block 321 at 0x10141000
EVICT: YES. evict 0x1013d000 with 0x10142000 to map block 322
EVICT: YES. evict 0x1013e000 with 0x10143000 to map block 323
EVICT: YES. evict 0x10003000 with 0x1013e000 to map block 318
EVICT: YES. evict 0x10140000 with 0x10144000 to map block 324
EVICT: YES. evict 0x10141000 with 0x10145000 to map block 325
EVICT: YES. evict 0x10142000 with 0x10146000 to map block 326
EVICT: YES. evict 0x10143000 with 0x10147000 to map block 327
EVICT: YES. evict 0x1013e000 with 0x10148000 to map block 328
EVICT: YES. evict 0x10144000 with 0x1013e000 to map block 318
EVICT: YES. evict 0x10145000 with 0x10149000 to map block 329
EVICT: YES. evict 0x10146000 with 0x1014a000 to map block 330
EVICT: YES. evict 0x10147000 with 0x1014b000 to map block 331
EVICT: YES. evict 0x10148000 with 0x1014c000 to map block 332
EVICT: YES. evict 0x1013e000 with 0x1014d000 to map block 333
EVICT: YES. evict 0x10149000 with 0x1013e000 to map block 318
EVICT: YES. evict 0x1014a000 with 0x1014e000 to map block 334
EVICT: YES. evict 0x1014b000 with 0x1014f000 to map block 335
EVICT: YES. evict 0x1014c000 with 0x1014b000 to map block 331
EVICT: YES. evict 0x1014d000 with 0x10150000 to map block 336
EVICT: YES. evict 0x1013e000 with 0x10151000 to map block 337
EVICT: YES. evict 0x1014e000 with 0x1013e000 to map block 318
EVICT: YES. evict 0x1014f000 with 0x10152000 to map block 338
EVICT: YES. evict 0x1014b000 with 0x10153000 to map block 339
EVICT: YES. evict 0x10150000 with 0x1014b000 to map block 331
EVICT: YES. evict 0x10151000 with 0x10154000 to map block 340
EVICT: YES. evict 0x1013e000 with 0x10155000 to map block 341
EVICT: YES. evict 0x10152000 with 0x1013e000 to map block 318
EVICT: YES. evict 0x10153000 with 0x10156000 to map block 342
EVICT: YES. evict 0x1014b000 with 0x10157000 to map block 343
EVICT: YES. evict 0x10154000 with 0x1014b000 to map block 331
EVICT: YES. evict 0x10155000 with 0x10158000 to map block 344
EVICT: YES. evict 0x1013e000 with 0x10159000 to map block 345
EVICT: YES. evict 0x10156000 with 0x1013e000 to map block 318
EVICT: YES. evict 0x10157000 with 0x1015a000 to map block 346
EVICT: YES. evict 0x1014b000 with 0x1015b000 to map block 347
EVICT: YES. evict 0x10158000 with 0x1014b000 to map block 331
EVICT: YES. evict 0x10159000 with 0x1015c000 to map block 348
EVICT: YES. evict 0x1013e000 with 0x1015d000 to map block 349
EVICT: YES. evict 0x1015a000 with 0x1013e000 to map block 318
EVICT: YES. evict 0x1015b000 with 0x1015e000 to map block 350
EVICT: YES. evict 0x1014b000 with 0x1015f000 to map block 351
EVICT: YES. evict 0x1015c000 with 0x1014b000 to map block 331
EVICT: YES. evict 0x1015d000 with 0x1013d000 to map block 317
EVICT: YES. evict 0x1013e000 with 0x10141000 to map block 321
EVICT: YES. evict 0x1015e000 with 0x1013e000 to map block 318
EVICT: YES. evict 0x1015f000 with 0x10142000 to map block 322
EVICT: YES. evict 0x1014b000 with 0x10143000 to map block 323
EVICT: YES. evict 0x1013d000 with 0x10144000 to map block 324
EVICT: YES. evict 0x10141000 with 0x10145000 to map block 325
EVICT: YES. evict 0x1013e000 with 0x10146000 to map block 326
EVICT: YES. evict 0x10142000 with 0x1013e000 to map block 318
EVICT: YES. evict 0x10143000 with 0x10147000 to map block 327
```

```
EVICT: YES. evict 0x10144000 with 0x10148000 to map block 328
EVICT: YES. evict 0x10145000 with 0x10149000 to map block 329
EVICT: YES. evict 0x10146000 with 0x1014a000 to map block 330
EVICT: YES. evict 0x1013e000 with 0x1014b000 to map block 331
EVICT: YES. evict 0x10147000 with 0x1014c000 to map block 332
EVICT: YES. evict 0x10148000 with 0x1013e000 to map block 318
EVICT: YES. evict 0x10149000 with 0x1014d000 to map block 333
EVICT: YES. evict 0x1014a000 with 0x1014e000 to map block 334
EVICT: YES. evict 0x1014b000 with 0x1014f000 to map block 335
EVICT: YES. evict 0x1014c000 with 0x1014b000 to map block 331
EVICT: YES. evict 0x1013e000 with 0x10150000 to map block 336
EVICT: YES. evict 0x1014d000 with 0x1013e000 to map block 318
EVICT: YES. evict 0x1014e000 with 0x10151000 to map block 337
EVICT: YES. evict 0x1014f000 with 0x10152000 to map block 338
EVICT: YES. evict 0x1014b000 with 0x10153000 to map block 339
EVICT: YES. evict 0x10150000 with 0x1014b000 to map block 331
EVICT: YES. evict 0x1013e000 with 0x10154000 to map block 340
EVICT: YES. evict 0x10151000 with 0x1013e000 to map block 318
EVICT: YES. evict 0x10152000 with 0x10155000 to map block 341
EVICT: YES. evict 0x10153000 with 0x10156000 to map block 342
EVICT: YES. evict 0x1014b000 with 0x10157000 to map block 343
EVICT: YES. evict 0x10154000 with 0x1014b000 to map block 331
EVICT: YES. evict 0x1013e000 with 0x10158000 to map block 344
EVICT: YES. evict 0x10155000 with 0x1013e000 to map block 318
EVICT: YES. evict 0x10156000 with 0x10159000 to map block 345
EVICT: YES. evict 0x10157000 with 0x1015a000 to map block 346
EVICT: YES. evict 0x1014b000 with 0x1015b000 to map block 347
EVICT: YES. evict 0x10158000 with 0x1014b000 to map block 331
EVICT: YES. evict 0x1013e000 with 0x1015c000 to map block 348
EVICT: YES. evict 0x10159000 with 0x1013e000 to map block 318
EVICT: YES. evict 0x1015a000 with 0x1015d000 to map block 349
EVICT: YES. evict 0x1015b000 with 0x1015e000 to map block 350
EVICT: YES. evict 0x1014b000 with 0x1015f000 to map block 351
EVICT: YES. evict 0x1015c000 with 0x1014b000 to map block 331
large file is good
EVICT: YES. evict 0x1013e000 with 0x1013d000 to map block 317
No runnable environments in the system!
Welcome to the JOS kernel monitor!
Type 'help' for a list of commands.
```