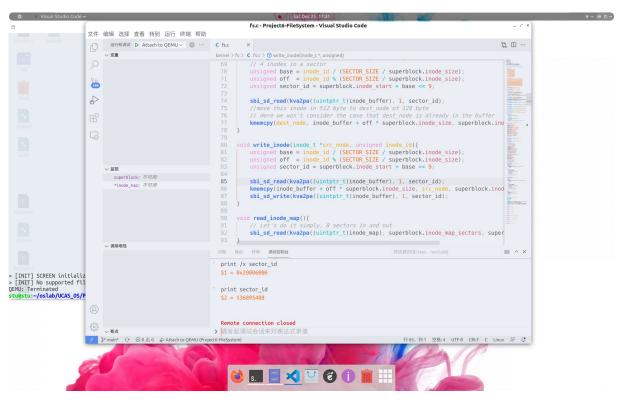
操作系统研讨课 实验报告

代瀚堃 2019K8009929051

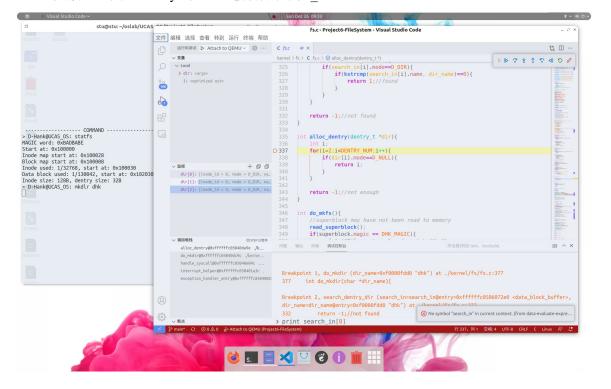
一、实验中遇到的问题

1. 读写 SD 卡出现 block error

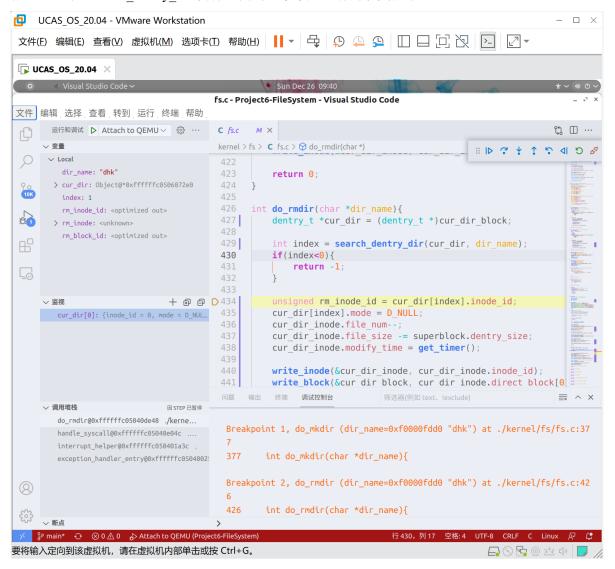
左移的优先级出错,导致块号变得很大:



2. 表项或者 inode 中的某些项没有初始化 mkdir 失败,原因是 dentry 的 mode 被初始化为 I DIR:



在 rmdir 时, search dentry dir 没有查找到,本来应该有的项变成了 NULL:



可能是 init dir block 有问题,看了一下,确实在初始化的时候把类型弄错了:

```
//NOTICE: this dir_block should be 4K-sized

void init_dir_block(dentry_t *dir_block, unsigned dir_id, unsigned parent_id){
    //here id is inode id
    //clear this block for directory
    kmemset(dir_block, 0, NORMAL_PAGE_SIZE);
    dir_block[0].inode_id = dir_id;

dir_block[0].mode = I_DIR;

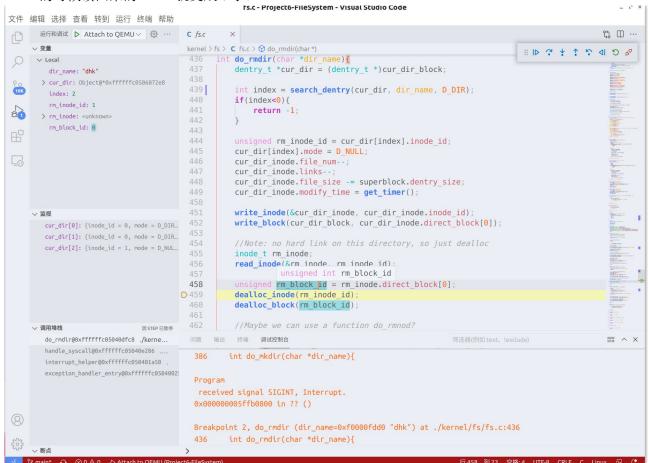
dir_block[0].name[0] = '.';
    dir_block[0].name[1] = '\0';

dir_block[1].inode_id = parent_id;
    dir_block[1].mode = I_DIR;
    dir_block[1].name[0] = '.';
    dir_block[1].name[0] = '.';
    dir_block[1].name[1] = '.';
    dir_block[1].name[2] = '\0';
}
```

应该是 D DIR

ls 卡死,问题出在 init_inode 这里,其步骤为: 开机-mkdir dhk-rmdir dhk-mkdir dhk-关机-开机-ls 第一次 mkdir: 根目录第 2 项, inode=1, block=1,

rmdir 的时候读回来的 block 就变成 0 了:

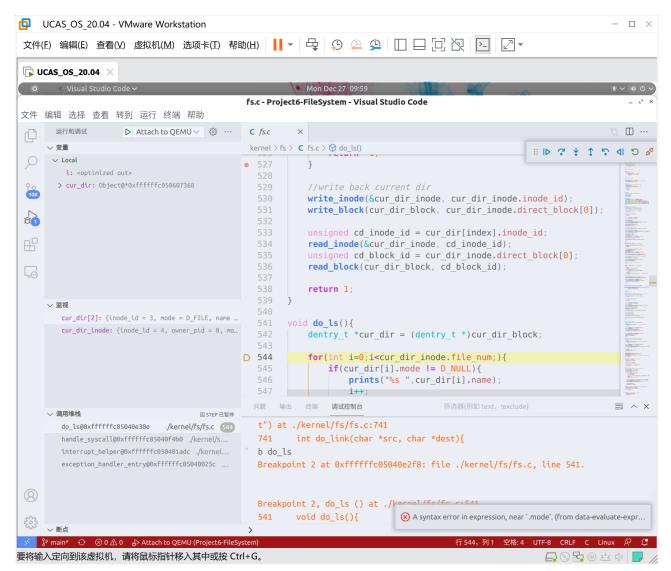


原因是在原来 mkdir 时,没有把 block 号存进去,写到磁盘上的还是 inode temp 内随机的数据:

```
unsigned new_block = alloc_block();
403
404
          //write new dir inode
405
          inode_t inode_temp;
406
          //read inode(&inode temp, new inode);
          init_inode(&inode_temp, new_inode, cur_dir_inode_inode_id, I_DIR, O_RDWR);
407
408
          write_inode(&inode_temp, new_inode);
409
410
          cur_dir[i].inode_id = new_inode;
411
          kstrcpy(cur_dir[i].name, dir_name);
          cur_dir[i].mode = D_DIR;
```

init_inode 只完成了部分初始化的内容,其余内容(file_size, file_num, block)需要根据创建的是文件还是目录来确定

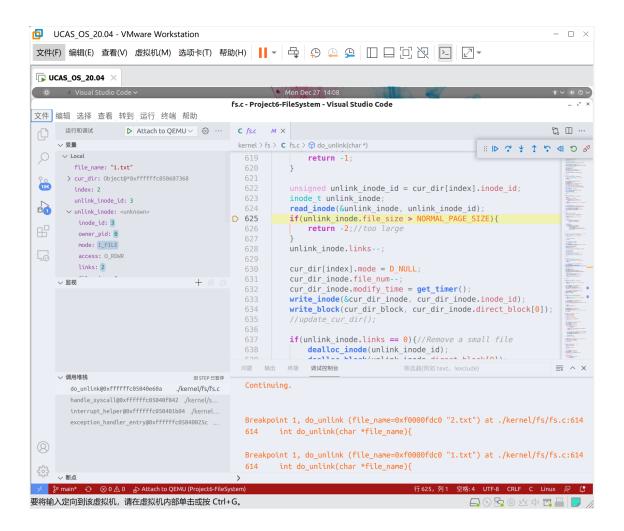
3. 块内的数据已经持久化到 SD 卡上,但没有同步到内存中的缓存 link 之后 ls 查看 test 下并没有 2.txt:



原因是 link 更新的是 inode_temp,并写到了 block 内,而缓存在 mem 中的当前目录并没有更新,所以每次改写当前目录后,都用 update cur dir 函数重新读一下当前目录的 inode 和数据块

rm 之后没有删除干净,其过程为:

创建目录/dhk/ucas, 创建文件/dhk/ucas/1.txt, 返回至根目录, 创建目录/test, 切至 test, 创建链接 ln /dhk/ucas/1.txt 2.txt, 删除 2.txt, 返回至 ucas 目录下删除 1.txt, 然后 statfs, 占用块仍为 5 个 原因是没有把 unlink node 写回, 后面读上来又是 2 了:



二、还有待解决的问题

- 1. 虽然我们已经实现了路径解析的函数 get_parent_dir,但目前只有 do_link 函数应用上,支持相对路径和绝对路径的查找,其他函数内没有调用这个函数,所以暂时只能支持相对路径
- 2. 由于缓冲区有限, cat 命令只支持一个 direct 块内文件内容的显示
- 3. do unlink 函数目前只支持删除一个 direct 块的文件,如果超过这一大小,会报错