OS Lab5 文档

515030910211姜子悦

Exercise1

在env\_create中增加对fs环境的特殊处理，使它能使用IO

jos采用了以一个特殊的env，fs env作为文件系统的方式，所有对文件系统的调用都是通过IPC的方式，让这个特殊env来处理。

Exercise2

实现了bc\_pgfault，来完成通过缺页异常来把一个page读进mem的操作。

实现了flush\_block，完成保证cache和disk一致性的函数。Jos对页的读写是修改mem中对应的cache，而flush\_block可以将这个cache刷回disk来真正完成写。

Exercise3

对照free\_block实现alloc\_block，alloc\_block就是从bitmap中找一个空block，把它标记为0，并且清空数据后返回一个block。

值得注意的是注释中提到的要记得flush bitmap。

这个和cse lab基本上差不多，只是jos的整体实现简化了inode。

Exercise4

实现这个文件系统需要我们来实现的最主要的部分，file\_block\_walk负责找到一个给定文件某个file block所在的slot，可能是一个direct的data block，有可能是一个indirect的。由于jos同样也只设计一级indirect block，这个部分和cse的lab基本上也是一样的。

而file\_get\_block，则用file\_block\_walk找到的slot，来返回或alloc一个data block。

Exercise5

Jos文件系统的ipc调用方式如guide图所示，我们负责实现serve和的devfile即可，其他都帮我们实现好了。

这个部分先实现read。

Serve的read就是按注释做一些判断以后，直接调用帮我们实现好的file read即可，file read会调用我们写好的block操作去写文件。

而devfile的read，也很简单。把给的参数放进ipc调用结构中，然后调用ipc函数来read即可。

Exercise6

基本同上，把读改成写就行了

注意这次是在devfile步骤检查buf长度是否爆掉。而read的时候是在serve中检查的。

Exercise7

实现fd\_alloc，这是我们ics cse课闻名已久的fd操作。和c中打开一个文件，会返回一个文件描述符fd给我们一样。这个函数会打开文件并给我们一个fd句柄来操作文件。值得注意的是返回值需要fd2num，而不是直接返回文件结构。

Exercise8

本来看上来讲spawn，io感觉这部分会很恐怖，感谢已经全部实现完了。

我们只需要实现一下系统调用就行了。

这个之前的lab实现过很多个了，获取env结构，然后设置相应属性即可。

要记得在handler中注册这个函数。

Question

1. 差不多1天？早上7点开始做，中间去上了一节2小时的课+半小时吃饭，下午7点做完。中间断断续续有些打断思路的事情。

Challenge

这次challenge感觉很难，没有一个会写的。cache的eviction policy有点思路，记录一下每个cache的最近使用时间，然后LRU策略去替换即可。然而并没有实现。