#### 1 FSLAB

### 1.1 实验目标

完成一个运行在用户态的文件系统,对模拟的块设备进行读写,从而实现一些基本的文件读写以及管理功能。

为了完成本实验,建议深入了解VSFS(简单文件系统)的结构设计,同时建议在实验开始前根据你需要读写的块大小/数量,需要实现的函数功能等来合理调整CSAPP中的*inode*设计,并阅读下发文件中的fuse.pptx,这将有助于你理解需要实现函数的功能。

#### 1.2 实验步骤

- 登陆服务器 ics.ruc.rvalue.moe
- 使用 cp ~/../fslab-handout.tar ~/ 将实验文件复制到自己的目录下
- 使用 tar xvf fslab-handout.tar 命令解压

disk.c: 实现一个模拟的块设备,请勿修改此文件

disk.h: 定义了你操作模拟块设备的接口函数,以及一些你可能用到的宏

fs.c: 你需要完成以及提交的文件

Makefile: make命令需要的脚本文件,请勿修改此文件

- 设计并完成你的文件系统
- 编译并测试你的文件系统
  - 通过 make 或 make debug 编译,如果编译通过,你的文件系统在debug模式下运行,此时整个程序保持前台运行,并输出对应的调试信息(你所写的 printf)。这个模式下输入 ctrl+c 会退出。但是千万不要输入 ctrl+z ,你将无法进行后续的编译或调试工作
  - 通过 make mount 编译并运行,此时你的文件系统以后台模式运行,相当于你的文件系统和其他系统组件一样真正被挂载。请记得使用 make umount 停止你的程序
  - 当你的文件系统运行时,你可以进入mnt/目录下进行操作,测试你的文件系统。如果你运行在debug模式下,你可以另外开一个shell/终端窗口进行测试
  - 当你以make mount 挂载你的文件系统后,你可以通过在 fslab-handout 文件夹下(同级目录),使用 ./traces/0.sh ,即可测试0号测试点,你可以通过与 0.ans 进行对比知道自己是否正确完成,后续最终测试我们将以自动化测试的方法完成,你可以自己运行脚本进行测试
- 提交fs.c以及实验报告到obe

- 80% (5%\*16) 正确性, 20%报告
- 在报告中,必须描述你的块设备组织结构,文件信息节点的结构,实现函数的重要细节,遇到的问题及解决方案,其余你认为重要的也请添加在报告中
- 描述不清/没有对结构进行描述的,将影响你的正确性分数

# 2 实验说明

## 2.1 需要完成的函数

#### int mkfs();

文件系统的初始化函数,写入文件系统基本信息以及根目录信息[/],如果一切正常返回0, 否则返回一个非0值。

函数	功能
fs_getattr	查询一个目录文件或常规文件的信息
fs_readdir	查询一个目录文件下的所有文件
fs_read	对一个常规文件进行读操作
fs_mkdir	创建一个目录文件
fs_rmdir	删除一个目录文件
fs_unlink	删除一个常规文件
fs_rename	更改一个目录文件或常规文件的名称(或路径)
fs_truncate	修改一个常规文件的大小信息
fs_utime	修改一个目录文件或常规文件的时间信息
fs_mknod	创建一个常规文件
fs_write	对一个常规文件进行写操作
fs_statfs	查询文件系统整体的统计信息
fs_open	打开一个常规文件
fs_release	关闭一个常规文件
fs_opendir	打开一个目录文件
fs_releasedir	关闭一个目录文件

表1其余需要实现的功能函数

# 2.2 关于块设备的信息和操作

你的文件系统运行在一个大小为**256MB**的虚拟块设备上,该块设备的访问粒度(块大小)为**4096**字节。你可以使用*disk.h*中的宏定义来提升你的代码风格。

你可以通过以下的两个函数实现读写操作。

```
int disk_read(int block_id, void *buffer);
int disk_write(int block_id, void *buffer);
```

- 参数 block id 表示你需要进行读写的块编号(从0开始,最大65535)
- 参数 buffer 指向一块大小为4096字节的连续内存。当进行读操作后,指定块中的数据会被读取到 buffer 指向的内存中。当进行写操作后, buffer 指向的内存中的数据会被写入指定块中
- 函数正常返回,返回值0,否则返回1

同时dish.h中提供了你可能用得到的宏定义:

• BLOCK\_SIZE: 一个块的大小, 4KB

• BLOCK\_NUM: 整个虚拟块设备包含的块数

• DISK\_SIZE: 整个虚拟块设备大小, 256MB

# 3 功能要求

- 至少250MB的真实可用空间
- 至少支持32768个文件及目录
- 不必支持相对路径([.]或[..])以及链接
- 只需支持文件名最大长度为24字符,且同一目录下不存在相同名字的文件或目录
- 只需支持单个文件最大8MB
- 只需支持单用户单线程访问,不需要考虑权限问题
- 虽然使用的是文件模拟块设备,但是不要尝试*mmap*等方法尝试将文件映射到内存,或通过接口之外提供的方法修改/拆分文件

# 4 温馨提示

- 本实验难度较大,请你留足充分的时间进行测试
- 你可以循序渐进的完成每个函数,每完成一个函数使用对应的命令测试
- 推荐在写操作之前完成读操作。但由于无法写入就无法测试读取,你可以通过在初始化的时候预先写入一些文件节点测试你的读操作
- 在每次运行你的程序后,你可以在 vdisk/ 目录下找到模拟块设备对应的文件,每个块大小都是4096字节,你可以通过 fread 函数将其中数据读取并分析
- 推荐使用debug模式进行调试,以确保你的命令运行在你所编写的文件系统上
- 推荐将一些常用操作(例如寻址)总结为函数/宏定义,提升你的代码复用率,这将作为代码风格占用一小部分分数
- 在完成一个操作时,要考虑如果出现错误,应该如何回到这步操作进行之前的状态
- 由于实现的文件系统出现了 bug 然后崩了,导致 VSCode ssh 无法连接,此时可以使用 其他工具 ssh 进入服务器,进入文件夹,执行 fusermount -zu mnt ,可能还需要删除 .vscode-server 文件夹。