FSLab

实验目标

完成一个运行在用户态的文件系统（FUSE），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。

实验步骤

- 登陆服务器
  - ics.ayaya.in
- 使用 `cp ~/../fslab-handout.tar ~/` 将实验文件复制到自己的用户目录下
- 使用 `tar xvf fslab-handout.tar` 命令解压
  - `disk.c`：实现了一个模拟的块设备，请不要修改这个文件
  - `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
- 设计并完成你的文件系统
- 提交**代码**以及**实验报告**到 obe

实验说明

需要完成的函数

```
int mkfs();
```

文件系统的初始化函数（格式化），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

函数的具体信息可以参照 PPT。

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 256MiB 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。你可以通过提供给你的以下两个函数来实现读写操作。

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

- 参数 `block_id` 表示你要进行读或写的块编号（从 0 开始，最大值为 65535）。
- 参数 `buffer` 指向一端大小为 4096 字节的连续内存。当进行读操作时，执行函数后，指定块中的数据会被读取到 `buffer` 指向的内存中；当进行写操作时，执行函数后，`buffer` 指向的内存中的数据会被写入到指定块中。
- 函数返回值表示函数运行的状态，如果一切正常返回 0，否则返回 1。

同时 `disk.h` 中定义了一些你可能用到的常量：

- `BLOCK_SIZE` 一个块的大小，4KiB
- `BLOCK_NUM` 整个虚拟块设备所包含的块数
- `DISK_SIZE` 整个虚拟块设备的大小，256MiB

功能要求

- 至少 250MiB 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（「.」和「..」）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名称相同的文件或目录
- 只需支持单个文件最大 8MiB，如果有能力可以支持更大的单文件大小
- 只需支持单用户单线程访问，不必考虑权限问题
- 虽然我们使用文件模拟块设备，但请不要尝试使用 `mmap` 等方法将文件映射到内存，或通过提供的接口之外的方法修改文件

编译并测试你的文件系统

- 通过 `make` 或 `make debug` 命令编译，如果编译通过，你的文件系统运行在 debug 模式，此时这个程序会保持前台运行状态，并输出对应的调试信息（你所写的 `printf` 语句），这个模式下输入 `ctrl+c` 命令会退出。警告：千万不要输入 `ctrl+z` 命令，你将无法进行后续的编译或调试工作！
- 通过 `make mount` 命令编译并运行，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它系统一样被真正挂载。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 putty / shell / 窗口进行测试。

温馨提示

- 你可以循序渐进地完成各个函数，每完成一个函数使用对应的命令进行测试
- 推荐你在写操作前完成读操作，但由于无法写入就无法测试读取，你可以通过在格式化的时候预先写入一些文件节点来测试你的读操作
- 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，每个的大小都是 4096 字节，你 可以通过 `fread` 函数将其中的数据读取出来进行分析
- 推荐使用 debug 模式进行测试和调试，以确保你的命令运行在你所编写的文件系统上
- 推荐将一些常用的操作归纳总结为函数，提高代码复用率，比如根据路径寻址等
- 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

可能遇到的问题

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。

评分标准

你需要提交**实验报告**和**代码**。不要抄袭。

- 代码（70%）
  - 多个测试点
  - 通过所有测试点可获得全部分数
- 报告（30%）
  - 块设备的组织结构
  - 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