哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

开课学期：2020 年秋季学期

课程名称：操作系统

实验名称：简单文件系统的设计与实现

实验性质：设计型

实验时间：12.10 地点：T2210

学生班级：1801105 班

学生学号：180110505

学生姓名：胡聪

评阅教师：

报告成绩：

实验与创新实践教育中心印制

2020 年 10 月

1. 实验目的

以 Linux 系统中的 EXT2 文件系统为例，熟悉该文件系统内部数据结构的组织方式和基本处理流程，在此基础上设计并实现一个简单的文件系统。

2. 实验环境

Linux

Microsoft Visual Code

3. 实验内容

3.1 实验任务

仅列出必做部分

（1） 实现青春版 Ext2 文件系统

功能包括：

1. 创建文件/文件夹（数据块可预分配）；
2. 读取文件夹内容；
3. 复制文件；
4. 关闭系统；
5. 系统关闭后，再次进入该系统还能还原出上次关闭时系统内的文件部署。

（2） 为实现的文件系统实现简单的 shell 以及 shell 命令以展示实现的功能。

1. ls - 展示读取文件夹内容
2. mkdir - 创建文件夹
3. touch - 创建文件
4. cp - 复制文件
5. shutdown - 关闭系统

3.2 实验过程

（1） 首先对 disk.c 进行了解

disk.c 是本次实验提供的虚拟磁盘接口，可以看出，这个文件模拟了一个大小为 4MB 的磁盘。

```
inline int get_disk_size()
{
        return 4*1024*1024;
```

```
}
```

提供的一些接口：

```
static int create_disk()
```

用于创建磁盘，将一个文件名为 disk 的 4MB 文件用 0 填满，进行初始化；

```
int open_disk()
```

用于打开磁盘，即读取 disk 文件中的内容；

```
int disk_read_block(unsigned int block_num, char* buf)
```

读磁盘块，注意这里要与后面会用到的数据块、超级块区分开；

DEVICE_BLOCK_SIZE，即每一个磁盘块的大小为 512B，而后面的数据块大小为 1024KB（1MB），即每个数据块占用 2 个磁盘块；

```
int disk_write_block(unsigned int block_num, char* buf)
```

与上面相反，写入磁盘块是将 buf 中的内容写到磁盘块中，同样 buf 的大小为 512B；

```
int close_disk()
```

关闭磁盘。

（2） 整体设计结构

shell.c —— 实现简单的 shell

block.c —— 数据块相关操作

command.c —— 对于输入 cmd type 的处理

dir_item.c —— 目录项相关操作

disk.c —— 磁盘接口

init_fs.c —— 文件系统初始化

inode.c —— inode 相关操作

path.c —— 对于输入 cmd path 部分的处理

utils.c ——常用接口封装

（3） 实验原理的再理解

1. Ext2 文件系统

指导书中把文件系统的基本布局简化为：超级块、inode 数组、数据块数组。

其中：

超级块：记录系统的整体信息。

```
typedef struct super_block
{
    int32_t magic_num;         // 幻数
    int32_t free_block_count; // 空闲数据块数
    int32_t free_inode_count; // 空闲 inode 数
    int32_t dir_inode_count;  // 目录 inode 数
    uint32_t block_map[128];  // 数据块占用位图
    uint32_t inode_map[32];   // inode 占用位图
} sp_block;
```

超级块记录文件系统的一些信息，分析一下占用大小可以发现，超级块占用字节为（32+32+32+32+32*128+32*32）/8 = 656B，超出了一个磁盘块的大小，为了我们处理起来比较方便，选择了让一个超级块占用 2 个磁盘块。

block_map 的每一个元素大小为 int32，而每一个比特代表一个数据块的使用情况，所以一共可以记录 128*32 个数据块的使用情况。

inode_map 同理，可以记录 32*32 个 inode 索引节点的占用情况。

inode 数组：对于每一个文件，对应一个 inode 结构体，记录文件大小、文件类型、连接数、数据块指针。

```
typedef struct inode
{
    uint32_t size;           // 文件大小
    uint16_t file_type;      // 文件类型（文件/文件夹）
    uint16_t link;           // 连接数
    uint32_t block_point[6]; // 数据块指针
} inode;
```

看得出来一个 inode 结构体的大小为 32B，而前面讲过，一个数据块的大小为 1024B，因此我们在这里定义两个量：

inode_map_index = inode_number / 32;

bit_index = inode_number % 32;

这两个变量用于记录位图信息，inode_map_index 是 inode_map 的索引，而 bit_index 是用于修改每一比特的 0/1 大小。

block_point 大小为 6 个 int，那么这说明在文件系统中如果使用直接索引方式，文件大小不能超过 6KB。

文件和文件夹 dir_item：

这里为了简化整个系统的表示，实际上目录和文件的区别仅仅在于 dir_item 和

inode 中的 type 不同，也不涉及文件内容的修改，因此创建文件和文件夹的逻辑是类似的。

```
typedef struct dir_item
{                       // 目录项一个更常见的叫法是 dirent(directory entry)
    uint32_t inode_id; // 当前目录项表示的文件/目录的对应 inode
    uint16_t valid;    // 当前目录项是否有效
    uint8_t type;      // 当前目录项类型（文件/目录）
    char name[121];    // 目录项表示的文件/目录的文件名/目录名
} dir_item;
```

可以计算出来一个 dir_item 的大小为 128B。

2. 对于 shell 的理解

shell 的实现实际上之前在写 xv6 实验的时候已经试过，第一个是要实现对于输入 cmd 的拆分，将其拆分为 type 和 path。

type 是指 shell 命令，这里包括 ls、mkdir、touch、cp、shutdown，对于其他命令应该给与报错提示。

对于 path 的拆分处理放在了后面（使用堆栈实现）。

（4） 具体函数设计实现（节省篇幅，仅写出了每个命令的执行该过程，其他的注释里都有写，这里就不再重复了）。

```
void shutdown_cmd()
{
    close_disk();
    exit(0);
}
```

shutdown 命令直接就是调用 close_disk()函数接口来实现了。

ls_cmd() //ls 命令

ls 命令的输入是路径，而要根据路径进行处理，则要将路径进行一下拆分。这里用到了一个自己定义的函数 search_dir_item_by_path()，从函数名理解，即根据路径找到 dir_item，然后根据 dir_item，也有了 inode_id（在 dir_item 中有记录），调用 read_inode()函数，可以找到对应的 inode，再一级一级查找到 dir_item（此过程每次有一个上一目录和当前目录，然后结构类似于栈，每次到下一级目录以后，上一级目录出栈）。

touch_cmd() //touch 命令

根据输入的路径，路径的最后一级就是所要创建的文件，而之前的是文件夹。首

先要对路径进行查找，如果路径不存在，则不能创建。创建文件与创建文件夹是类似的，创建新的 inode 和 dir_item 标记类型为 FILE 并更新之即可。

mkdir_cmd() //mkdir 命令

检测创建的目录是否为根目录，是根目录提示不能创建（第一次可以创建），不是根目录的话，找到上一级文件夹的 dir_item 和 inode，然后为需要创建的文件夹创建对应的 dir_item。

cp_cmd() //cp 命令

cp 命令是复制命令。这里首先要提到，之前在 shell 中 cmd 定义为二维数组，这样就可以方便对复制文件的原文件和复制后文件的路径进行管理，当然写报告的时候也想到了通过对空格进行检测的方式，也还是可以实现路径的分离的。

4. 总结及实验课程感想

本次实验耗时很长，写得也算是心力交瘁，但是还是学习到了很多东西，首先是对 C 语言有了更深入的了解，还有就是文件系统这一块也有了更加深入的认识。但是通过这次实验，还是有很多建议想提供给老师和助教，希望能够给课程带来一些帮助。个人仅仅是站在一名成绩偏下的学生的角度来说的，因此这些建议必然有很多不合理的地方，还请见谅。

（1） 首先是前 4 个实验，感觉指导书的内容读完之后还是很难看懂实验该做些什么内容，而本学期很多同学是同时选修了密码学基础、编译原理等专业选修课的，多个实验叠加在一起就很容易让人不想去从头完成实验，而是“参考”网络上（博客、GitHub 等）的实验资源，而非去认认真真完整读一遍原版指导书（何况还是英语的，对于不少同学来说还是存在困难），所以如果下一届同学还是写 xv6 实验的话，希望能够提供更为详尽的指导说明；

（2） 然后是文件系统实验，感觉这个实验设计得非常好，能够锻炼能力，而且主观上也愿意去写这个实验，实验指导书也做得非常详细，但是问题还是该实验与计算机网络实验相重叠，导致写起来很累（也许对于大佬来说是很简单的事情，但是对于我这样的学生来说，对于很多知识点掌握不熟悉，就要花费特别多的时间，计网和操作系统实验都是）。主要是希望课程安排能够再合理一些。

最后感谢本课程老师、助教的辛苦付出。