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

# 实验报告

开课学期：2020 秋季学期

课程名称：操作系统

实验名称：实验五：简单文件系统的设计与实现

实验性质：设计型

实验时间：2020.12.13 地点：

学生班级：1801105

学生学号：180110527

学生姓名：李秋阳

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

语言及标准：c11

运行环境：Ubantu 64-bit

编译器：Clion 2019.2.5

# 3.实验内容

## 3.1 实验任务

### （1）实现青春版Ext2文件系统

系统结构参考ext2系统即可，不需要严格按照ext2设计，但必须是模拟文件系统，并能完成如下功能：

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

可参考实现以下shell命令：

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

对于一个物理磁盘，如果将其格式化为Ext2文件系统，则按照Ext2的规范可以磁盘块组织为超级块、组描述符、索引节点、位图等用于管理数据，剩余的空间用于文件数据的保存。在本次实验中，只需要实现简化的Ext2文件系统（元数据区只包括超级块和inode数组），布局如图（来自实验指导书）所示：


我们需要将超级块、inode和数据块等数据结构进行实现，并通过对这几个主要数据结构的操作来完成文件系统功能的实现。

首先要做的是文件系统的初始化，也就是磁盘的初始化。我们需要调用虚拟磁盘接口来打开磁盘，读入磁盘块。然后对根目录结构体进行初始化，分两次磁盘读写读取大小为1k的超级块。此时，我们可以读取到超级块的幻数来判断超级块是否已经经过初始化，如果没有的话则需要对超级块和根目录进行初始化。

```
//幻数为0，第一次进入系统，初始化超级块和根目录；不为0则直接获取根目录为当前目录
if(magic_num == 0) {
    printf("Disk is already format.\nWelcome EXT2 file system...\n");
    super_block_ent_root_init();
} else {
    printf("Welcome EXT2 file system...\n");
    struct inode inode_root;
    disk_read_block(2, disk_rw_buf);
    buffer_read(disk_rw_buf, &inode_root, INODE_INDEX, INODE_SIZE);
    current_inode = inode_root;
    current_inode_index = INODE_INDEX;
    current_level = 1;
    strcpy(current_path[0], "root");
}
```

对超级块和根目录的初始化需要对根目录的inode进行初始化，然后记录根目录为当前文件系统的工作路径。在根目录下，我们需要创建并写入“.”和“..”的目录信息，再设置好数据块占用位图和inode占用位图的相关信息，初始化超级块的幻数、空闲数据块数（初始为4*1024-3）和空闲inode数（初始为1024-1）。

```
//初始化超级块，0、1、64分别是超级块、inode数组、数据块的数据索引
inode_map_set(dir_ent_root.inode_id,1);
block_map_set(0,1);
block_map_set(1, 1);
block_map_set(64, 1);

super_block.dir_inode_count = 1;
super_block.magic_num = MAGIC_NUMBER;
super_block.free_block_count = MAX_BLOCK_NUM - 3;
super_block.free_inode_count = 1024 - 1;
super_block_write();
```

对于各种操作指令的实现，核心是要编写遍历各种数据结构的函数，能够找到空闲的block、inode、dir_item等，对于创建文件、文件夹和复制等命令，还需要找到指定目录下是否存在已经存在文件或文件夹，如果存在则可以复制，不存在才可以创建。寻找空闲dir_item和判断文件是否存在的遍历匹配需要读入指定的磁盘块，将数据读到缓存区，然后在缓存区中进行操作。

在需要创建文件或文件夹时，如果找到了当前目录项有数据块空闲且目录项也空闲，则可以进行创建，创建的目录项即为当前空闲目录项，如果有数据块空闲但目录项不空闲，则直接分配一个新的数据块给要创建的文件或文件夹。

```
    //如果数据块和当前目录都已经用完，则退出
    if (block_point_id == 6 && dir_ent.valid == 1) {
        fprintf(stderr,"Current directory has no more space\n");
        return;
    }
    //数据块没用完但目录项不空闲，分配一个新的数据块
    else if (block_point_id < 6 && dir_ent.valid==1) {
        int block_id = block_ergodic();
        current_inode.block_point[block_point_id] = block_id;
```

```
        block_map_set(block_id, 1);
        super_block.free_block_count --;
        disk_read_block(block_id * 2, disk_rw_buf);
        super_block.free_inode_count --;
        inode_map_set(free_inode, 1);
        dir_ent.inode_id = free_inode;
        dir_ent.type = FILE_TYPE_DIR;
        strcpy(dir_ent.name, path[0]);

        buffer_write(disk_rw_buf, &dir_ent, 0, DIR_SIZE);
        disk_write_block(block_id * 2, disk_rw_buf);
    }
    //数据块没用完，目录项为空闲目录项
    else {
        int block_id = current_inode.block_point[block_point_id];
        super_block.free_inode_count--;
        inode_map_set(free_inode, 1);
        dir_ent.inode_id = free_inode;
        dir_ent.type = FILE_TYPE_DIR;
        strcpy(dir_ent.name, path[0]);
        dir_ent.valid = 1;
        buffer_write(disk_rw_buf, &dir_ent, disk_pos*DIR_SIZE, DIR_SIZE);
        disk_write_block(block_id*2 + disk_blk_part, disk_rw_buf);
    }
```

# 4. 总结及实验课程感想

在操作系统以前的各个课程的实验课对我来说大多是理解容易，写代码需要额外下点功夫，但是操作系统的几次实验都需要在理解上花费很大的功夫，特别是xv6的alloc、buddy等实验，虽然不用写多少代码，但是如果不理解原理也不知道从哪里下手。本次简化版Ext2文件系统的实验也是一样，完成实验的关键在于理解Ext2文件系统的格式。

本次实验的另外一个难点在于由于数据结构较多，调试比较复杂，虽然我用了功能比较强大的IDE来进行调试，但是在一些小问题上依然花费了很多时间。其实主要原因还是因为对是实验中的数据结构理解不够到位，没有意识到什么时候这个变量不能动，那个参数需要赋值 。

比较遗憾的是本次实验因为实验期间其他的课程、实验和考试等缘故，没有足够的时间来优化代码，提高代码的健壮性，导致了很多函数都有代码的冗余，错误检测也不够完善，希望以后能多留一点时间来给学生完成。

# 用户手册

包括ls，mkdir，touch，cp，cd，shutdown指令

第一次进入：

```
Disk is already format.
Welcome EXT2 file system...
==>mkdir test
==>ls
./
../
test/
==>touch test.c
==>ls
./
../
```

```
test/
test.c
==>cp test.c
==>ls
./
../
test/
test.c
test.c_1
==>cd test
root/test/==>ls
./
../
==>mkdir test1
==>touch test1.c
==>ls
./
../
test1/
test1.c
==>shutdown
Exit EXT2 file system, thanks for using.
```

退出后进入：

```
Welcome EXT2 file system...
==>ls
./
../
test/
test.c
test.c_1
==>shutdown
Exit EXT2 file system, thanks for using.
```