Project 6 File System 设计文档

中国科学院大学
[姓名] 蔡洛姗
[日期] 2021/1/11

1. 文件系统初始化设计

（1）请阐述你设计的文件系统对磁盘的布局（可以使用图例表示），包括从磁盘哪个位置开始，superblock、inode map、block 或 sector map、inode table 以及数据区各自占用的磁盘空间大小

```
Disk layout:
[ boot block | kernel image ] (512MB)
[ Superblock(512B) | block map(16KB) | inode map(512B) | inode
blocks(256KB) | data blocks ] (512MB)
```

（2）请列出你设计的 superblock 和 inode 数据结构，并阐明各项含义。请说明你设计的文件系统能支持的最大文件大小，最多文件数目，以及单个目录下能支持的最多文件/子目录数目。

```
typedef struct superblock {
    uint32_t magic;                 // identify the file system
    uint32_t size;                  // Size of file system image (sectors)
    uint32_t start_sector;          // 在磁盘中的起始扇区号

    uint32_t blockmap_offset;       // block map 的起始地址 by sector
    uint32_t blockmap_num;          // 所占扇区数

    uint32_t inodemap_offset;       // block map 的起始地址 by sector
    uint32_t inodemap_num;          // 所占扇区数

    uint32_t inodes_offset;
    uint32_t inodes_num;

    uint32_t datablock_offset;
    uint32_t datablock_num;

    uint32_t inode_sz;              // 每个 inode 的大小
    uint32_t dentry_sz;             // 每个 dentry 的大小
}superblock_t;

typedef struct inode {
    uint32_t inum;
    uint8_t  mode;                  // Inode mode: FILE/DIR
```

```
    uint8_t  access;              // File mode: R/W/RW
    uint16_t ref;                 // link num
    uint32_t size;                // Size of file (bytes)
    uint32_t valid_size;           // used data/dentry bytes
    uint32_t direct_bnum[NDIRECT];  // Direct data block addresses
    uint32_t indirect1_bnum;       // First indirect address
    uint32_t indirect2_bnum;       // Double indirect address
    uint64_t mtime;               // modified time
}inode_t;
```

目前我的文件系统能支持 8*4096B 大小的文件，一个目录下最多可容纳 4096/32=128 项文件/子目录。

2. 文件操作设计

（1）请说明创建一个文件所涉及的元数据新增和修改操作，例如需要新增哪些元数据，需要修改哪些元数据

创建一个文件需要为新文件分配一个 inode，初始化 inode 中的内容并分配一个 block 给新文件，修改父目录的修改时间和大小，修改 inodemap 和 blockmap 使用情况。

3. 目录操作设计

（1）请说明文件系统执行 ls 命令查看一个绝对路径时的操作流程

```
void do_cd(uintptr_t dirname){
    // check if FS exist
    uintptr_t status;
    status = sbi_sd_read(SB_MEM_ADDR, 1, START_SECTOR);
    superblock_t *superblock = (superblock_t *)(SB_MEM_ADDR+P2V_OFFSET);
    if(superblock->magic != MAGIC_NUM){
        prints("[ERROR] No File System!\n");
        return;
    }
    // look up the path
    inode_t *ionde_temp=current_inode;
    char *path = (char *)dirname;
    if(path[0]!='\0'){
        if(find_path(path,T_DIR)==0 || current_inode->mode != T_DIR){
            prints("[ERROR] PATH NOT FOUND!\n");
            current_inode = ionde_temp;
            return;
        }
    }
}

static    char head_dir[30];
static    char tail_dir[30];
```

```
int find=0;
int dep=0;
//递归调用，查找每一级的目录并进入
int find_path(char * path, int type){
    inode_t *inode_temp = current_inode;
    if(dep==0){
        find=0;
    }

    /* Absolute path */
    if(path[0]=='/'){
        current_inode=(inode_t *)(INODE_MEM_ADDR+P2V_OFFSET);
        if(strlen(path)==1)
            return 1;
        int m;
        for(m=0;m<strlen(path);m++)
            path[m]=path[m+1];
    }

    int i,j;
    for(i=0,j=0;i<strlen(path) && path[i]!='/';i++,j++){
        head_dir[j]=path[i];
    }
    head_dir[j++]='\0';
    i++;

    for(j=0;i<strlen(path);i++,j++)
        tail_dir[j]=path[i];
    tail_dir[j]='\0';

    if(head_dir[0]=='\0')
                return 0;

    inode_t *ip;
    if((ip = dirlookup(current_inode,head_dir,type)) != 0){
        current_inode = ip;
        if(tail_dir[0]=='\0'){
            find=1;
                    return 1;
        }
        dep++;
        find_path(tail_dir,type);
    }
```

```
    if(find==0){
        current_inode = inode_temp;
        dep=0;
        return 0;
    }else{
        dep=0;
        return 1;
    }
}
```