OS lab 文件系统实验报告

151220012 陈永恒

实验要求

- 内核：实现文件系统的基本操作，并提供系统调用 open, close, read, write,lseek；
- 库：封装上述系统调用；
- 工具：格式化磁盘镜像的 formatter，从宿主机拷贝文件至磁盘镜像中指定目录的copy2myfs,从磁盘镜像按文件名读文件的read_myfs；
- 游戏：将游戏所需资源（如图片等）作为独立文件存放，利用新增库函数读取资源文件，保存游戏进度或记录等。

实验进度

- 内核：实现了文件系统的基本操作，提供系统调用 open, close, read,write, lseek，接口和linux标准接口一致
- 库：封装了上述系统调用，分别为 fs_open, fs_close, fs_read,fs_write, fs_lseek
- 工具：全部实现，源代码在lab的根目录，使用方法也可以直接运行程序得到
- 游戏：因为我的游戏界面都是自己画出来的，没有有到外部的资源，所以我只是game.bin当作一个文件copy到我的系统中
- 实现了多级索引(支持大文件)

实验分析

我的镜像文件分布如下

```
/*
+------------+--------+------+------+------+-------+-------+
--------+
| bootloader | bitmap |  dir  | inode | data  |  ...  |
  ...  |
+------------+--------+------+------+------+-------+-------+
--------+
1 sec           64 sec  2 sec   32 sec
*/
```

sec表示一个block(sector),即512个字节.
对应的结构是参照讲义的:

```
typedef struct
{
    unsigned int blocks[128];
}inode;

inode inodes[32];
typedef struct {
    char     filename[24];
    unsigned int file_size;
    unsigned int inode_offset;

}dirent;

dirent dir[512 / sizeof(dirent)*2];

unsigned char bitmap[512 * 64];
```

即inode包含文件物理块的序号，目录项包含文件名，文件大小和其所对应的inode，bitmap就是一个位矢图

在规划了镜像的分布后，几个小工具就很容易实现了，具体不细说

剩下的就是open等函数的实现了. 首先定义FCB:

```
typedef struct FCB
{
    FILE_FLAG flag;
    int inuse;
    int fd;
    unsigned int inode_offset;
    unsigned int offset;
    struct FCB* next;
}FCB;
```

按照讲义的建议，我使用单向链表的方法来管理每个进程打开的文件， flag成员则是记录了打开文件的方式(读或写)

为了读写文件系统，最重要的是准确定位文件在磁盘的位置，而这个在format后的镜像中，通过目录项和inode的帮助下很容易实现．具体实现参考代码

为了支持大文件，重要的还是要找到对于的物理块的索引．我写了一个辅助函数:

```
unsigned int big_file_block_offset(unsigned int block_offset,unsigned int next_block_num){
    unsigned int paddr=block_to_address(next_block_num);
    inode tmp;
    readseg((unsigned char*)tmp.blocks,SECTSIZE,paddr);
    if(block_offset<511){
        return tmp.blocks[block_offset];
    }else{
        return big_file_block_offset(block_offset-511,tmp.blocks[511]);
    }

}
```

这个函数在inode中块的位移超过了127时调用，它把inode的最后一个物理块当作下一级的inode，如果还是没找到，则递归调用，否则返回对应的物理块．有了这个函数，把相应的read和write修改后就能支持大文本．（PS:这个大文本是在操作系统中支持的，小工具中的 copy2myfs并不支持把大文件拷贝到镜像中）

实验难点

这个实验我觉得最难的是write的实现，因为这个操作会导致文件的增长．我采用的方法是在写完的时候比较文件的偏移和文件大小，如果是前者大，就表示文件的大小增加了，如果当前的块能存放要写的内容，就直接更新文件大小，否则申请新的没有被使用的block,并更新文件大小和对应的inode.