南京大学
NANJING UNIVERSITY

Lab5 文件系统

助教：孙立志

实验内容

- 格式化程序:按照文件系统的数据结构，构建磁盘文件(即os.img)。
- 内核:初始化时，按照文件系统的数据结构，读取磁盘，加载用户程序，并对 OPEN 、 READ 、 WRITE 、 LSEEK 、 CLOSE 、REMOVE 、STAT这些系统调用提供接口
- 库:对上述系统调用进行封装
- 用户:对上述库函数进行测试
- 具体要求参照网站第1节

文件控制块

- 内核使用 FCB（File Control Block，文件控制块）这一数据结构对进程打开的文件进行管理
- FCB 中需要记录其对应的是哪个文件，该文件是以哪种方式打开的（读、写），该文件的读写偏移量
- FCB 的索引号称为文件描述符，每个进程的PCB 中对其打开的文件的文件描述符进行记录

文件系统操作接口

- open 为 Linux 提供的系统原语，其用于打开（或创建）由路径 path 指定的文件，并返回文件描述符 fd， fd 为该文件对应的内核数据结构 FCB 的索引，参数 flags用于对该文件的类型、访问控制进行设置

  int open(char *path, int flags);

- read 为 Linux 提供的系统原语，其用于从 fd 索引的FCB 中的文件读写偏移量处开始，从文件中读取 size 个字节至从 buffer 开始的内存中，并返回成功读取的字节数，若文件支持 seek 操作，则同时修改该 FCB 中的文件读写偏移量

  int read(int fd, void *buffer, int size);

文件系统操作接口

- write 为 Linux 提供的系统原语，其用于向 fd 索引的 FCB 中的文件读写偏移量处开始，向文件中写入从buffer 开始的内存中的 size 个字节，并返回成功写入的字节数，若文件支持 seek 操作，则同时修改该 FCB中的文件读写偏移量

  int write(int fd, void *buffer, int size);

- lseek 为 Linux 提供的系统原语，其用于修改 fd 索引的 FCB 中的文件读写偏移量（若文件支持 seek 操作）

  int lseek(int fd, int offset, int whence);

文件系统操作接口

- close 为 Linux 提供的系统原语，其用于关闭由 fd索引的 FCB

  int close(int fd);

- remove 为 C 标准库的函数，其用于删除 path 指定的文件

  int remove(char *path);

- stat 为 Linux 提供的系统原语，其用于通过文件名filename获取文件信息，并保存在buffer所指的结构体stat中

  int stat(char *filename, struct stat *buffer);

文件、目录、设备、管道、套接字、链接

- 为统一用户操作接口，类 Unix 系统将物理硬件设备与内核提供的功能抽象为文件，并分配相应的 inode，例如以下代码，通过向标准输出的设备文件中写入 Hello World!，即可实现在当前终端中打印出该字符串

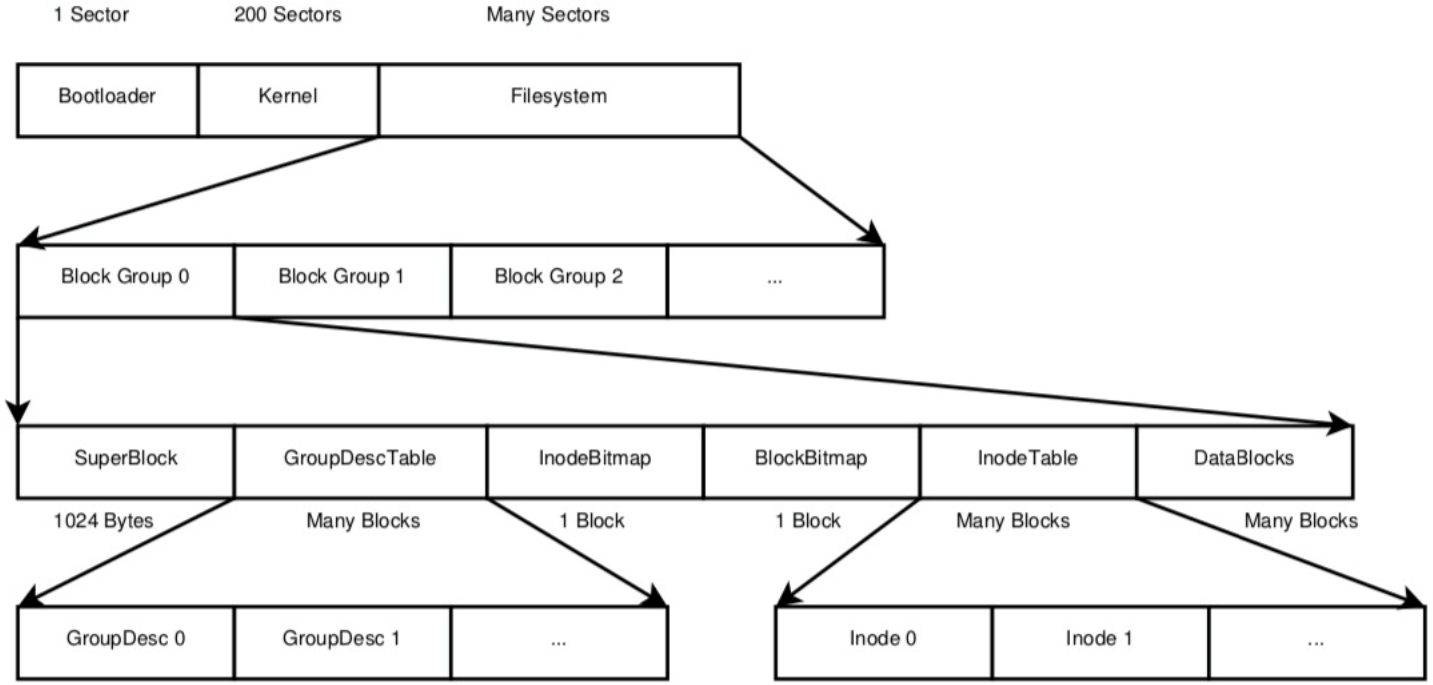
```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int main (int argc, char *argv[]) {
        int fd = open("/dev/stdout", O_WRONLY);
        write(fd, (void*)"Hello␣World!\n", 13);
        close(fd);
        return 0;
}
```

文件、目录、设备、管道、套接字、链接

- 类 Unix 系统的文件系统通常将所有文件划分为通常文件（Regular File）、目录文件（Directory）、块设备文件（Block Device）、字符设备文件（Character Device）、管道文件（FIFO）、套接字文件（Socket）、链接文件（Symbolic Link）
- 块设备文件与字符设备文件是按照存取方式进行的划分，前者例如/dev/sda、/dev/loop0，后者例如/dev/tty1、/dev/random，前者支持随机访存，后者仅支持顺序访存，一般看来，前者支持 Seek 操作，后者不支持 Seek 操作
- 设备文件同样可以按照其是否具有物理实体进行划分，即物理设备（对实际存在的物理硬件的抽象，例如/dev/sda）与虚拟设备（内核提供的功能，例如/dev/loop0）

磁盘文件布局

- 0 号扇区为主引导扇区（MBR）， 1 到 200 号扇区存储内核程序， 201 号开始的扇区按照文件系统的数据结构进行格式化，用戶程序也依照文件系统的数据结构写入其中



文件系统数据结构

- 文件系统的布局仿照 EXT4 文件系统，将扇区划分为 Block进行管理（例如连续两个扇区作为一个 Block），连续的Block 又进一步划分为 Block Group
- 每个 Block Group 在头部为 Meta Block，记录文件系统的元数据，依次包含 Super Block、 Group Descriptor Table、Inode Bitmap、 Block Bitmap； Meta Block 之后则为 InodeTable 以及真正存储通常文件以及目录文件的 Data Block
- Super Block 中记录了整个文件系统的信息，例如总扇区数，Inode 总数，Block 总数，可用 Inode 数，可用 Block 数等Group Descriptor Table 中则记录了各个 Block Group 的信息，例如可用 Inode 数，可用 Block 数
- Inode Bitmap 以及 Block Bitmap 则分别用于对该 Block Group 中 Inode Table 以及 Data Block 的使用情况进行记录

文件系统目录结构

- 文件系统的目录结构如下所示，本次实验要求编写格式化程序对其进行构建

```
+/
|---+sbin
|    |---init            #用户态初始化程序
|    |---...
|---+dev
|    |---stdin           #标准输入设备文件
|    |---stdout          #标准输出设备文件
|    |---...
|---+usr
|    |---...             #用户文件
|---...
```

作业提交

- 实验框架代码的发布和提交均在课程网站cslabcms.nju.edu.cn。PPT和实验指导文件也会发布在此
- 截止时间:2024-6-16 23:55
- 如果你无法完成实验，可以选择不提交，作为学术诚信的奖励，你将会获得10%的分数；但若发现抄袭现象，抄袭双方(或团体)在本次实验中得0分，后续可能有其他惩罚
- 本实验的最终解释权由助教所有