

实验 4 指导

【任务】

为 Linux 系统设计一个简单的二级文件系统。要求做到以下几点。

(1) 可以实现下列几条命令：

login	用户登录
dir	列目录
create	创建文件
delete	删除文件
open	打开文件
close	关闭文件
read	读文件
write	写文件

(2) 列目录时要列出文件名、物理地址、保护码和文件长度。

(3) 源文件可以进行读写保护

【程序设计】

(1) 设计思想

本文件系统采用两级目录,其中第一级对应于用户账号,第二级对应于用户账号下的文件。另外,为了简单,本文件系统未考虑文件共享、文件系统安全以及管道文件与设备文件等特殊内容。对这些内容感兴趣的读者,可以在本系统的程序基础上进行扩充。

(2) 主要数据结构

① 索引节点

struct inode

```
{
    struct inode * i_forw;
    struct inode * i_back;
    char i_flag;
    unsigned int i_ino;           /* 磁盘索引节点标号 */
    unsigned int i_count;        /* 引用计数 */
    unsigned short di_number;    /* 关联文件数,当为 0 时,则删除该文件 */
    unsigned short di_mode;     /* 存取权限 */
    unsigned short di_uid;      /* 磁盘索引节点用户 id */
    unsigned short di_gid;      /* 磁盘索引节点组 id */
    unsigned int di_addr [NADDR]; /* 物理块号 */
}
```



② 磁盘索引节点

```
struct dinode
{
    unsigned short di_number;    /* 关联文件数 */
    unsigned short di_mode;      /* 存取权限 */

    unsigned short di_uid;
    unsigned short di_gid;
    unsigned long di_size;       /* 文件大小 */
    unsigned int di_addr[NADDR]; /* 物理块号 */
}
```

③ 目录项结构

```
struct direct
{
    char d_name[DIRSIZ];    /* 目录名 */
    unsigned int d_ino;      /* 目录号 */
}
```

④ 超级块

```
struct filsys
{
    unsigned short s_isize;    /* 索引节点块数 */
    unsigned long s_fsize;     /* 数据块块数 */
    unsigned int s_nfree;      /* 空闲块块数 */
    unsigned short s_pfree;    /* 空闲块指针 */
    unsigned int s_free [NICFREE]; /* 空闲块堆栈 */

    unsigned int s_ninode;     /* 空闲索引节点数 */
    unsigned short s_pinode;   /* 空闲索引节点指针 */
    unsigned int s_inode [NICINOD]; /* 空闲索引节点数组 */
    unsigned int s_rinode;     /* 铭记索引节点 */

    char s_fmod;               /* 超级块修改标志 */
};
```

⑤ 用户密码

```
struct pwd
{
    unsigned short p_uid;
    unsigned short p_gid;
    char password [PWOSIZ];
}
```



```
};
```

⑥ 目录

```
struct dir
{
    struct direct direct [DIRNUM];
    int size;
};
```

⑦ 查找内存索引节点的 hash 表

```
struct hinode
{
    struct inode *i_forw;
};
```

⑧ 系统打开表

```
struct file
{
    char f_flag; /* 文件操作标志 */
    unsigned int f_count; /* 引用计数 */
    struct inode *f_inode; /* 指向内存索引节点 */
    unsigned long f_off; /* 读/写指针 */
};
```

⑨ 用户打开表

```
struct user
{
    unsigned short u_default_mode; /* 用户标志 */
    unsigned short u_uid; /* 用户组标志 */
    unsigned short u_gid; /* 用户打开文件表 */
    unsigned short u_ofile [NOFILE];
};
```

(3) 主要函数

- | | |
|------------|--------------|
| ① iget() | 索引节点内容获取函数 |
| ② iput() | 索引节点内容释放函数 |
| ③ mkdir() | 目录创建函数 |
| ④ namei() | 目录搜索函数 |
| ⑤ balloc() | 磁盘块分配函数 |
| ⑥ bfree() | 磁盘块释放函数 |
| ⑦ ialloc() | 分配索引节点区函数 |
| ⑧ ifree() | 释放索引节点区函数 |
| ⑨ iname() | 搜索当前目录下文件的函数 |
| ⑩ access() | 访问控制函数 |



- | | |
|-------------|------------|
| ⑪ _dir() | 显示目录和文件用函数 |
| ⑫ chdir() | 改变当前目录用函数 |
| ⑬ open() | 打开文件函数 |
| ⑭ create() | 创建文件函数 |
| ⑮ read() | 读文件用函数 |
| ⑯ write() | 写文件用函数 |
| ⑰ login() | 用户登录函数 |
| ⑱ logout() | 用户退出函数 |
| ⑲ format() | 文件系统格式化函数 |
| ⑳ install() | 进入文件系统函数 |
| ㉑ close() | 关闭文件函数 |
| ㉒ halt() | 退出文件系统函数 |
| ㉓ delete() | 文件删除函数 |

以上函数的详细描述略。

(4) 主程序说明

Begin

- Step1 对磁盘进行格式化
- Step2 调用 install(), 进入文件系统
- Step3 调用 _dir(), 显示当前目录
- Step4 调用 login(), 用户注册
- Step5 调用 mkdir() 和 chdir() 创建目录
- Step6 调用 create(), 创建文件 0
- Step7 分配缓冲区
- Step8 写文件 0
- Step9 关闭文件 0 并释放缓冲区
- Step10 调用 mkdir() 和 chdir() 创建子目录
- Step11 调用 create(), 创建文件 1
- Step12 分配缓冲区
- Step13 写文件 1
- Step14 关闭文件 1 并释放缓冲区
- Step15 调用 chdir 将当前目录移到上一级
- Step16 调用 create(), 创建文件 2
- Step17 分配缓冲区
- Step18 调用 write(), 写文件 2
- Step19 关闭文件 2 并释放缓冲区
- Step20 调用 delete(), 删除文件 0
- Step21 调用 create(), 创建文件 3
- Step22 为文件 3 分配缓冲区
- Step23 调用 write(), 写文件 3
- Step24 关闭文件 3 并释放缓冲区
- Step25 调用 open(), 打开文件 2
- Step26 为文件 2 分配缓冲区



Step27 写文件 3 后关闭文件 3

Step28 释放缓冲区

Step29 用户退出 (logout)

Step30 关闭 (halt)

End

由上述描述过程可知,该文件系统实际是为用户提供一个解释执行相关命令的环境。主程序中的大部分语句都被用来执行相应的命令。

下面给出每个过程的相关 C 语言程序。读者也可以使用这些子过程,编写出一个用 Shell 控制的文件系统界面。

【程序】

(1) 编程管理文件 makefile

本文件系统程序用 makefile 编程管理工具进行管理。其内容如下:

```
/* makefile */
filsys: main.o igetput.o iallfre.o ballfre.o name.o access.o log.o close.o
      create.o delete.o dir.o dirlt.o open.o rdwt.o format.o install.o halt.o
cc-o filsys main.o igetput.o iallfre.o ballfre.o name.o access.o log.o close.o
      create.o delete.o dir.o dirlt.o open.o rdwt.o format.o install.o halt.o
main.o: main.c filsys.h
      cc-c main.c
igetput.o: igetput.c filsys.h
      cc-c igetput.c
iallfre.o: iallfre.c filsys.h
      cc-c iallfre.c
ballfre.o: ballfre.c filsys.h
      cc-c ballfre.c
name.o: name.c filsys.h
      cc-c name.c
access.o: access.c filsys.h
      cc-c access.c
log.o: log.c filsys.h
      cc-c log.c
close.o: close.c filsys.h
      cc-c close.c
create.o: create.c filsys.h
      cc-c create.c
delete.o: delete.c filsys.h
      cc-c delete.c
dir.o: dir.c filsys.h
      cc-c dir.c
dirlt.o: dirlt.c filsys.h
      cc-c dirlt.c
open.o: open.c filsys.h
      cc-c open.c
```



```
rdwt.o: rdwt.c filesys.h
    cc-c rdwt.c
format.o: format.c filesys.h
    cc-c format.c
install.o: install.c filesys.h
    cc-c install.c
halt.o: halt.c
    cc-c halt.c
```

(2) 头文件 filesys.h

头文件 filesys.h 用来定义本文件系统中所使用的各种数据结构和常数。

/* filesys.h 定义本文件系统中的数据结构和常数 */

```
#define BLOCKSIZ 512
#define SYSOPENFILE 40
#define DIRNUM 128
#define DIRSIZ 14
#define PWDSIZ 12
#define PWDNUM 32
#define NOFILE 20
#define NADDR 10
#define NHINO 128
#define USERNUM 10
#define DINODESIZ 32

/* filsys */
#define DINODEBLK 32
#define FILEBLK 512
#define NICFREE 50
#define NICINOD 50
#define DINODESTART 2 * BLOCKSIZ
#define DATASTART (2 + DINODEBLK) * BLOCKSIZ

/* di_mode */
#define DIEMPTY 0000

#define DIFILE 01000
#define DIDIR 02000

#define UDIREAD 00001 /* user */
#define UDIWRITE 00002
#define UDIEXECUTE 00004
#define GDIREAD 00010 /* group */
#define GDIWRITE 00020
#define GDIEXECUTE 00040
#define ODIREAD 00100 /* other */
```




```
#define ODIWRITE 00200
#define ODIEXECUTE 00400
```

```
#define READ 1
#define WRITE 2
#define EXECUTE 4
```

```
#define DEFAULTMODE 00777
```

```
/* i_flag */
```

```
#define IUPDATE 00002
```

```
/* s_fmod */
```

```
#define SUPDATE 00001
```

```
/* f_flag */
```

```
#define FREAD 00001
```

```
#define FWRITE 00002
```

```
#define FAPPEND 00004
```

```
/* error */
```

```
#define DISKFULL 65535
```

```
/* fseek origin */
```

```
#define SEEK_SET 0
```

```
/* 文件系统数据结构 */
```

```
struct inode{
```

```
    struct inode * i_forw;
```

```
    struct inode * i_back;
```

```
    char i_flag;
```

```
    unsigned int i_ino;
```

```
    unsigned int i_count;
```

```
    unsigned short di_number;
```

```
    unsigned short di_mode;
```

```
    unsigned short di_uid;
```

```
    unsigned short di_gid;
```

```
    unsigned int di_size;
```

```
    unsigned int di_addr[NADDR];
```

```
};
```

```
struct dinode{
```

```
    unsigned short di_number;
```

```
    unsigned short di_mode;
```

```
    unsigned short di_uid;
```

```
/* 磁盘索引节点标志 */
/* 引用计数 */
/* 关联文件数, 当为 0 时, 则删除该文件 */
/* 存取权限 */
```

```
/* 文件大小 */
/* 物理块号 */
```

```
/* 关联文件数 */
/* 存取权限 */
```



```

    unsigned short di_gid;
    unsigned long di_size;
    unsigned int di_addr[NADDR];
};

struct direct{
    char d_name[DIRSIZ];
    unsigned int d_ino;
};

struct filsys{
    unsigned short s_isize;
    unsigned long s_fsize;
    unsigned int s_nfree;
    unsigned short s_pfree;
    unsigned int s_free[NICFREE];

    unsigned int s_ninode;
    unsigned short s_pinode;
    unsigned int s_inode[NICINOD];
    unsigned int s_rinode;

    char s_fmod;

};

struct pwd{
    unsigned short p_uid;
    unsigned short p_gid;

    char password[PWOSIZ];
};

struct dir{
    struct direct direct[DIRNUM];
    int size;
};

struct hinode{
    struct inode * i_forw;
};

struct file{
    char f_flag;
    unsigned int f_count;
    struct inode * f_inode;
};

/* 文件大小 */
/* 物理块号 */

/* 索引节点块数 */
/* 数据块块数 */
/* 空闲块块数 */
/* 空闲块指针 */
/* 空闲块堆栈 */

/* 空闲索引节点数 */
/* 空闲索引节点指针 */
/* 空闲索引节点数组 */
/* 铭记索引节点 */

/* 超级块修改标志 */

/* 当前目录大小 */

/* hash 表指针 */

/* 文件操作标志 */
/* 引用计数 */
/* 指向内存索引节点 */

```




```

    unsigned long f_off;
}; /* 读/写指针 */

struct user {
    unsigned short u_default_mode;
    unsigned short u_uid;
    unsigned short u_gid;
    unsigned short u_ofile[NOFILE]; /* 用户打开文件表 */
};

/* 以下为全局变量 */
extern struct hinode hinode[NHINO];
extern struct dir dir; /* 当前目录 (在内存中全部读入) */
extern struct file sys_ofile[SYSOPENFILE];
extern struct filsys filsys; /* 内存中的超级块 */
extern struct pwd pwd[PWDNUM];
extern struct user user[USERNUM];
extern FILE * fd; /* the file system column of all the system */
extern struct inode * cur_path_inode;
extern int user_id, file_block;

/* prototype of the sub routine used in the file system */
extern struct inode * iget();
extern input();
extern unsigned int balloc();
extern bfree();
extern struct inode * ialloc(); /* you want to format the disk? */
extern ifree();
extern unsigned int namei();
extern unsigned int iname(); /* format will erase all content on disk */
extern unsigned int access();
extern _dir();
extern mkdir();
extern chdir();
extern dirlt();
extern unsigned short open();
extern create();
extern unsigned int read();
extern unsigned write();
extern int login();
extern logout();
extern install();
extern format();
extern close();
extern halt();

```

