实验三 文件系统

1.1 实验目的

通过一个简单文件系统的设计，加深理解文件系统的内部实现原理。

1.2 实验内容及完成情况

模拟 EXT2 文件系统原理设计实现一个类 EXT2 文件系统。

(1) 列目录时列出文件类型、文件名、创建时间、最后访问时间和修改时间；

(2) 切换路径时既可以**进入当前目录下的子目录**、也可以**通过相对路径**或者**绝对路径**来切换路径；

(3) 删除目录时通过不同命令**可以选择只删除空目录**或者**递归删除目录下所有目录和文件**；

(4) 实现修改密码、登录登出等功能；

(5) 通过设置文件的访问权限对文件进行**读写保护**；

(6) 写入文件时仿照 linux 系统的 echo 命令，既可以**覆盖文件写入**也可以**在文件末尾追加内容**。

1.3 实验思想

为了进行简单的模拟，基于 Ext2 的思想和算法，设计一个类 Ext2 的文件系统，实现 Ext2 文件系统的一个功能子集。并且用现有操作系统上的文件来代替硬盘进行硬件模拟。

- 类 ext2 文件系统的数据结构

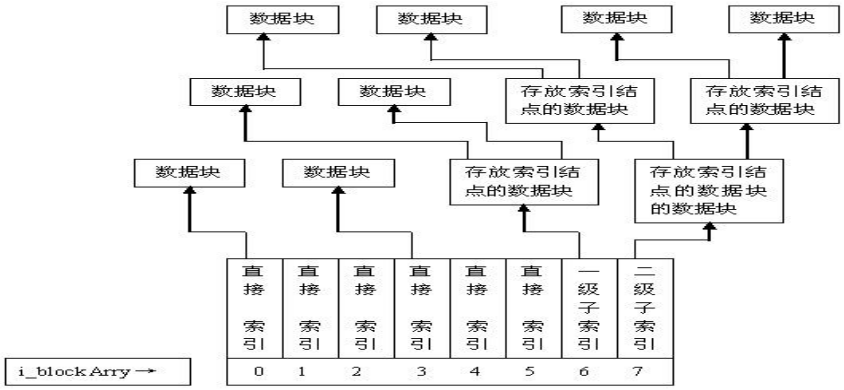
组描述符	数据块位图	索引结点位图	索引结点表	数据块
1 block	1 block	1 block	512 blocks	4096 blocks
512 Bytes	512 Bytes	512 Bytes	256 KB	2 MB

块的定义：为简单起见，逻辑块大小与物理块大小均定义为 512 字节。由于位图只占用一个块，因此，每个组的数据块个数以及索引结点的个数均确定为 512*8=4096。进一步，

每组的数据容量确定为 4096*512B=2MB。另外，模拟系统中，假设只有一个用户，故可以省略去文件的所有者 ID 的域。

组描述符：为简单起见，只定义一个组。因此，组描述符只占用一个块。同时，superblock 块省略，其功能由组描述符块代替，即组描述符块中需要增加文件系统大小，索引结点的大小，卷名等原属于 superblock 的域。

索引结点的数据结构中，采用多级索引机制。直接索引 6 块，一级索引 512/2=256 块，二级索引 256x256=65536 块。



目录与文件：与 ext2 相同，目录作为特殊的文件来处理。目录的索引结点指向的数据块中存储该目录下所有目录和文件的目录项，文件的索引结点指向的数据块中存储文件的具体内容。

- 类 ext2 文件系统程序设计

底层函数：完成所有文件系统底层的操作封装。并为上层即命令层提供服务。

命令层函数：文件系统所支持的命令及其功能在这一层实现。

用户接口层：主要功能是接收及识别用户命令，词法分析，提取命令及参数。组织调用命令层对应的命令实现相应功能。本层实际上是一个基于命令层基础上的 shell。

- 其他细节

文件和目录的三种时间：

CreateTime：目录或文件创建时的时间

LastAccessTime：上一次目录列出所有项　　文件读出的时间

ModifyTime：目录创建新文件或目录、删除子目录或文件　　文件写入的时间

1.4 实验步骤

(1) 定义数据结构

```
struct ext2_group_desc//组描述符 68 字节
struct ext2_inode //索引结点 64 字节
struct ext2_dir_entry//目录体 最大 261 字节 最小 7 字节
```

(1) 完成底层函数的编写

```
void update_group_desc() /*将内存中的组描述符更新到"硬盘"*/
void reload_group_desc() /*载入可能已更新的组描述符*/
void reload_inode_entry(uint16_t i,ext2_inode* inode_i)/*载入特定的索引结点*/
void update_inode_entry(uint16_t i,ext2_inode* inode_i)/*更新特定的索引结点*/
void update_block_bitmap()//更新 block 位图
void reload_block_bitmap()//载入 block 位图
void update_inode_bitmap()//更新 inode 位图
void reload_inode_bitmap()//载入 inode 位图
void reload_block_entry(uint16_t i,uint8_t* block)//将第 i 个数据块读到缓冲区 block 中
void update_block_entry(uint16_t i,uint8_t* block)//将缓冲区 block 的内容写入第 i 个数据块
uint16_t alloc_block()//分配一个数据块,返回数据块号
void free_block(uint16_t block_num)//根据传入的数据块号释放该数据块
uint16_t get_index_one(uint16_t i,uint16_t first_index)  /*返回一级索引的数据块号*/
uint16_t get_index_two(uint16_t i,uint16_t two_index)  /*返回二级索引的数据块号*/
void reload_block_i(uint16_t i,unsigned char tmp_block[512],ext2_inode tmp_inode)//载入当前索引节点第 i 个数据块中的某个目录项到缓冲区中
void reload_dir_i(uint16_t i,ext2_dir_entry* dir_i,uint16_t offset,ext2_inode tmp_inode)//将某个目录项写入当前索引结点的某个数据块中
void update_index_one(uint16_t i,uint16_t block_num,ext2_inode* tmp_inode)//将当前结点的第 i 个数据块号写入一级索引块
void update_index_two(uint16_t i,uint16_t block_num,ext2_inode* tmp_inode)//将当前结点的第 i 个数据块号写入二级索引块
void update_inode_newblock(uint16_t block_num,ext2_inode* tmp_inode)//将新分配的数据块记录到索引结点的信息中
void delete_inode_allblock(uint16_t inode_num)//删除该索引结点的所有数据块
uint16_t get_inode()//分配一个 inode,返回序号
void free_inode(uint16_t inode_num)//根据传入的索引节点号在索引节点表中释放该节点
void dir_prepare(ext2_inode* inode_i,uint16_t last,uint16_t cur)//新建目录时当前目录和上一级目录的初始化
uint16_t search_filename(char tmp[255],uint8_t file_type,ext2_inode* cur_inode,int flag)//在当前目录中查找文件或子目录，返回索引节点号
```

```
int test_fd(uint16_t inode_num)//在文件打开表中查找文件是否存在
```

- **多级索引的实现：**

通过遍历索引结点中的数据块，若 i<6 则为直接索引，直接通过访问索引节点的数据块指针数组找到相应的数据块号；若 6<=i<6+256 则为一级索引，通过索引节点的数据块指针数组的第 7 个元素（inode_block[6]）找到一级索引数据块，再通过数据块号在该数据块中顺序存储读出 2 字节数据块号，则找到相应的数据块；若 i>=6+256 则为一级索引，通过索引节点的数据块指针数组的第 8 个元素（inode_block[7]）找到二级索引第一级数据块，再通过 i 判断要寻找的数据块号所在的数据块的数据块号存在二级索引第一级数据块的哪个 2 字节的位置，从而找到二级索引第二级数据块，接着在这个数据块中寻找最终的数据块号。

如果是要为索引结点分配新的数据块，则需要新的数据块号是存在直接索引数组中、一级索引数据块中还是二级索引数据块中。还需判断分配前是否用到过多级索引，是否需要为一级索引或者二级索引分配索引数据块，并将数据块号存入其中。

(2) 完成初始化操作

```
void initialize_disk()
```

初始化组描述符，分配根目录的索引结点并为其分配一个数据块，为根目录创建当前目录和上级目录的目录项并存入相应的数据块中。

```
void initialize_memory()
```

初始化文件打开表并载入，载入组描述符和当前目录索引节点。

(3) 完成命令层函数的编写

```
void dir_ls()
```

列出当前目录下的所有项

```
void cd(char path_name[256])
```

进入当前目录下的子目录或者进入输入的相对路径或绝对路径

```
void create(uint8_t type_num,char name[255])
```

在当前目录下创建一个文件或者目录，完成在当前目录下插入新文件或目录的目录项，为

新文件或目录分配索引节点和数据块，对新目录进行初始化等操作。

```
void delete(uint8_t type_num,char name[255],int flag)
void delete_dir(uint16_t delete_inode_num)
void delete_file(uint16_t delete_inode_num)
```

删除当前目录下某个目录或文件，通过 flag 选择删除目录的方式（只删除空目录或递归删除），delete 函数调用 delete_dir 实现递归删除目录。

```
void open_file(unsigned char name[255])
```

打开文件操作，即将该文件的索引节点号存入文件打开表中。

```
void close_file(unsigned char name[255])
```

关闭文件操作，即将该文件的索引节点号从文件打开表中移出。

```
void read_file(char name[512])
```

读文件操作，读出该文件所有数据块中存储的内容，

```
void write_file(unsigned char name[512],int flag,char source[2560])
```

写入文件操作，通过 flag 选择是覆盖写入还是追加写入。若文件已分配的数据块不够，则根据需要分配新的数据块。

```
void attrib(unsigned char name[255],char rwx[6])
```

修改当前目录下某个文件或目录的访问权限

```
void help()     列出命令菜单
```

(4) 完成用户接口层函数的编写

```
void format()//初始化
int password()//修改密码
void quit()//退出文件系统
int login()//登入
void logoff()//登出
void shell()//启动用户接口
```

(5) 编写主函数

(6) 测试文件系统功能，具体结果见下一条内容

1.5 运行结果及功能测试

(1) 初始化及输入密码登陆

```
[root@localhost expfile]# ./file
Creating the ext2 file system
The ext2 file system has been installed!
Hello! Welcome to simulation of ext2 file system!
Please input the password: 123
Wrong password! Please try again: 666
[root@ /]#
```

初始密码为 666，若输入错误有一次重新输入密码的机会，两次输入都错误则退出文件系统

(2) 下面是通过 help 命令显示出的该文件系统可以实现的功能以及相应的命令

```
[root@ /]# help
   ********************************************************************************
   *                   An simulation of ext2 file system                          *
   *                                                                              *
   * The available commands are:                                                  *
   * 1.change dir   : cd+dir_name/path     2.create dir       : mkdir+dir_name    *
   * 3.create file  : mkf+file_name        4.delete empty dir : rmdir+dir_name    *
   * 5.delete file  : rm+file_name         6.delete all in dir: rmdir+-r+dir_name *
   * 7.open   file  : open+file_name       8.write  file      : write+file_name+>/>>+ *
   * 9.close  file  : close+file_name      10.read  file      : read+file_name    *
   * 11.list  items : ls                   12.this  menu      : help              *
   * 13.format disk : format               14.password        : password          *
   * 15.login       : login                16.logoff          : logoff            *
   * 17.change imode: chmod+name+rwx       18.exit            : quit              *
   ********************************************************************************
```

(3) ls：列出当前目录下所有项

```
[root@ /]# ls
mode    type        size      CreateTime                LastAccessTime            ModifyTime                name
r_w_x   Directory   17   B    Wed Nov 29 09:18:09 2023  Wed Nov 29 09:18:09 2023  Wed Nov 29 09:18:09 2023  .
r_w_x   Directory   17   B    Wed Nov 29 09:18:09 2023  Wed Nov 29 09:18:09 2023  Wed Nov 29 09:18:09 2023  ..
```

- 当前目录为根目录，且初始化后没有创建文件或目录，因此只有两个目录项。
- 目录的 size 表示目录下所有目录项的大小之和，文件的 size 表示文件内容的大小所占的字节数。
- 文件或目录新创建时默认权限都为 r_w_x，可以使用命令进行修改。由于此处并没有不同类型的用户，因此 imode 只设置了一组值。

(4) mkdir+dir_name：在当前目录下创建一个目录

```
[root@ /]# mkdir dir1
Directory dir1 is created successfully!
[root@ /]# ls
mode    type        size      CreateTime                LastAccessTime            ModifyTime                name
r_w_x   Directory   28   B    Wed Nov 29 09:18:09 2023  Wed Nov 29 09:18:12 2023  Wed Nov 29 09:20:24 2023  .
r_w_x   Directory   28   B    Wed Nov 29 09:18:09 2023  Wed Nov 29 09:18:12 2023  Wed Nov 29 09:20:24 2023  ..
r_w_x   Directory   17   B    Wed Nov 29 09:20:24 2023  Wed Nov 29 09:20:24 2023  Wed Nov 29 09:20:24 2023  dir1
```

成功创建目录 dir1，并且可以看到根目录的 modifytime 修改为了 dir1 的 createtime。

(5) mkf+file_name：在当前目录下新建一个文件

```
[root@ /]# mkf file1
File file1 is created successfully!
[root@ /]# ls
mode    type        size      CreateTime                LastAccessTime            ModifyTime                name
r_w_x   Directory   40   B    Wed Nov 29 09:18:09 2023  Wed Nov 29 09:20:26 2023  Wed Nov 29 09:30:24 2023  .
r_w_x   Directory   40   B    Wed Nov 29 09:18:09 2023  Wed Nov 29 09:20:26 2023  Wed Nov 29 09:30:24 2023  ..
r_w_x   Directory   17   B    Wed Nov 29 09:20:24 2023  Wed Nov 29 09:20:24 2023  Wed Nov 29 09:20:24 2023  dir1
r_w_x   File        0    B    Wed Nov 29 09:30:24 2023  Wed Nov 29 09:30:24 2023  Wed Nov 29 09:30:24 2023  file1
[root@ /]# mkf file1
There has been a file named file1!
[root@ /]# mkf dir1
There has been a directory named dir1!
```

- 成功创建文件 file1，并且再次 ls 可以看到 lastaccesstime 被修改为上一次使用 ls 命令访问根目录的时间
- 可以看出，在同一目录下不允许存在同名文件或同名目录，也不允许文件和目录同名。

下面进行一些文件和目录的创建以便进行后面的测试。

```
root/
├── dir1/
│   ├── dir11/
│   └── file11
├── file1
├── file2
├── dir2/
│   ├── dir21/
│   └── file_a
└── dir3
```

创建好后访问根目录如下：

```
[root@ /]# ls
mode    type        size      CreateTime                LastAccessTime            ModifyTime                name
r_w_x   Directory   74   B    Wed Nov 29 09:18:09 2023  Wed Nov 29 09:31:11 2023  Wed Nov 29 09:52:42 2023  .
r_w_x   Directory   74   B    Wed Nov 29 09:18:09 2023  Wed Nov 29 09:31:11 2023  Wed Nov 29 09:52:42 2023  ..
r_w_x   Directory   42   B    Wed Nov 29 09:20:24 2023  Wed Nov 29 09:20:24 2023  Wed Nov 29 09:54:40 2023  dir1
r_w_x   File        0    B    Wed Nov 29 09:30:24 2023  Wed Nov 29 09:30:24 2023  Wed Nov 29 09:30:24 2023  file1
r_w_x   File        0    B    Wed Nov 29 09:52:23 2023  Wed Nov 29 09:52:23 2023  Wed Nov 29 09:52:23 2023  file2
r_w_x   Directory   42   B    Wed Nov 29 09:52:32 2023  Wed Nov 29 09:52:32 2023  Wed Nov 29 09:55:13 2023  dir2
r_w_x   Directory   17   B    Wed Nov 29 09:52:42 2023  Wed Nov 29 09:52:42 2023  Wed Nov 29 09:52:42 2023  dir3
```

(6) cd+dir_name/path：改变当前目录路径

进入当前目录下的子目录 dir1：

```
[root@ /]# cd dir1
[root@ /dir1/]# ls
mode    type        size      CreateTime                LastAccessTime             ModifyTime                 name
r_w_x   Directory   42   B    Wed Nov 29 09:20:24 2023  Wed Nov 29 09:58:15 2023   Wed Nov 29 09:54:40 2023   .
r_w_x   Directory   74   B    Wed Nov 29 09:18:09 2023  Wed Nov 29 09:55:19 2023   Wed Nov 29 09:52:42 2023   ..
r_w_x   Directory   17   B    Wed Nov 29 09:54:21 2023  Wed Nov 29 09:54:21 2023   Wed Nov 29 09:54:21 2023   dir11
r_w_x   File        0    B    Wed Nov 29 09:54:40 2023  Wed Nov 29 09:54:40 2023   Wed Nov 29 09:54:40 2023   file11
```

返回上级目录，再使用相对路径进入 dir1 下的目录 dir11：

```
[root@ /dir1/]# cd ..
[root@ /]# cd dir1/dir11
[root@ /dir1/dir11/]# ls
mode    type        size      CreateTime                LastAccessTime             ModifyTime                 name
r_w_x   Directory   17   B    Wed Nov 29 09:54:21 2023  Wed Nov 29 09:54:21 2023   Wed Nov 29 09:54:21 2023   .
r_w_x   Directory   42   B    Wed Nov 29 09:20:24 2023  Wed Nov 29 10:02:59 2023   Wed Nov 29 09:54:40 2023   ..
```

在任意目录下使用绝对路径可进入另一路径下的目录：

```
[root@ /dir1/dir11/]# cd /dir2/dir21
[root@ /dir2/dir21/]# ls
mode    type        size      CreateTime                LastAccessTime             ModifyTime                 name
r_w_x   Directory   17   B    Wed Nov 29 09:55:07 2023  Wed Nov 29 09:55:07 2023   Wed Nov 29 09:55:07 2023   .
r_w_x   Directory   42_  B    Wed Nov 29 09:52:32 2023  Wed Nov 29 09:52:32 2023   Wed Nov 29 09:55:13 2023   ..
```

使用 cd /命令可直接返回根目录：

```
[root@ /dir2/dir21/]# cd /
[root@ /]#
```

(7) chmod+name+rwx：修改当前目录下名为 name 的目录或文件的访问权限，rwx 为包含想要设置权限字符的字符串

权限	普通文件	目录文件
r	可读取此文件的实际内容（cat查看文件）	具有读目录结构列表的权限
w	可编辑该文件的内容（vim，echo），但并不具备删除该文件本身的权限（删除文件由文件的上层目录控制，跟文件本身的权限无关。）	**具有删除、移动目录内文件的权限**
x	该文件具有可以被系统执行的权限	用户能否进入该目录（cd），chmod 同时有w和x权限才可以创建、删除文件和目录

- 测试 x 权限：

```
[root@ /]# chmod dir3 rw
Please input the password: 666
[root@ /]# cd dir3
No permission to access to the directory!
```

- 测试 w 权限：

```
[root@ /]# chmod dir3 rx
Please input the password: 666
[root@ /]# cd dir3
[root@ /dir3/]# mkdir dir33
No permission to create file/directory in current directory!
```

- 测试 r 权限：

```
[root@ /dir3/]# cd ..
[root@ /]# chmod dir3 wx
Please input the password: 666
[root@ /]# cd dir3
[root@ /dir3/]# ls
No permission to list items in current directory!
```

(8) rmdir+dir_name：删除空目录，若目录不为空则不能删除

```
[root@ /]# rmdir dir1
The block is not empty.
[root@ /]# rmdir dir
No such directory named dir!
[root@ /]# rmdir dir3
dir3 is deleted successfully!
[root@ /]# ls
mode    type        size      CreateTime                LastAccessTime            ModifyTime                name
r_w_x   Directory   63    B   Wed Nov 29 09:18:09 2023   Wed Nov 29 09:55:19 2023   Wed Nov 29 10:32:24 2023   .
r_w_x   Directory   63    B   Wed Nov 29 09:18:09 2023   Wed Nov 29 09:55:19 2023   Wed Nov 29 10:32:24 2023   ..
r_w_x   Directory   42    B   Wed Nov 29 09:20:24 2023   Wed Nov 29 10:02:59 2023   Wed Nov 29 09:54:40 2023   dir1
r_w_x   File        0     B   Wed Nov 29 09:30:24 2023   Wed Nov 29 09:30:24 2023   Wed Nov 29 09:30:24 2023   file1
r_w_x   File        0     B   Wed Nov 29 09:52:23 2023   Wed Nov 29 09:52:23 2023   Wed Nov 29 09:52:23 2023   file2
r_w_x   Directory   42    B   Wed Nov 29 09:52:32 2023   Wed Nov 29 09:52:32 2023   Wed Nov 29 09:55:13 2023   dir2
```

- dir1 不为空所以不能删除
- 当前目录下不存在 dir 所以不能删除
- dir3 为空目录被成功删除

(9) rmdir+-r+dir_name：删除目录及目录下所有目录和文件

```
[root@ /]# rmdir -r dir2
dir2 is deleted successfully!
[root@ /]# ls
mode    type        size      CreateTime                LastAccessTime            ModifyTime                name
r_w_x   Directory   52    B   Wed Nov 29 09:18:09 2023   Wed Nov 29 10:32:26 2023   Wed Nov 29 10:36:25 2023   .
r_w_x   Directory   52    B   Wed Nov 29 09:18:09 2023   Wed Nov 29 10:32:26 2023   Wed Nov 29 10:36:25 2023   ..
r_w_x   Directory   42    B   Wed Nov 29 09:20:24 2023   Wed Nov 29 10:02:59 2023   Wed Nov 29 09:54:40 2023   dir1
r_w_x   File        0     B   Wed Nov 29 09:30:24 2023   Wed Nov 29 09:30:24 2023   Wed Nov 29 09:30:24 2023   file1
r_w_x   File        0     B   Wed Nov 29 09:52:23 2023   Wed Nov 29 09:52:23 2023   Wed Nov 29 09:52:23 2023   file2
```

(10) open+file_name：打开文件，close+file_name：关闭，read+file_name：读出

write+file_name+>/>>+...：>为写入时覆盖原文件，>>为写入时追加内容到文件末尾

- 测试打开文件和读文件

```
[root@ /]# read file1
File file1 has not been opened!
[root@ /]# open file1
File file1 is opened successfully!
[root@ /]# read file1

File file1 is empty!
```

- 文件未打开时不能进行读写操作
- 成功打开 file1，未写入时读取文件显示文件为空

➢ 测试两种写入文件的方式

```
[root@ /]# write file1 > 123
[root@ /]# read file1

123
```

```
[root@ /]# write file1 >> abc
[root@ /]# read file1

123abc
```

➢ 测试写入过程中数据块的分配及文件的访问时间和修改时间

```
[root@ /]# write file1 > aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
[root@ /]# read file1

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

[root@ /]# write file1 >> bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
[root@ /]# read file1

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

```
[root@ /]# ls
mode   type       size    CreateTime                 LastAccessTime             ModifyTime                 name
r_w_x  Directory  52   B  Wed Nov 29 09:18:09 2023   Wed Nov 29 10:36:29 2023   Wed Nov 29 10:36:25 2023   .
r_w_x  Directory  52   B  Wed Nov 29 09:18:09 2023   Wed Nov 29 10:36:29 2023   Wed Nov 29 10:36:25 2023   ..
r_w_x  Directory  42   B  Wed Nov 29 09:20:24 2023   Wed Nov 29 10:02:59 2023   Wed Nov 29 09:54:40 2023   dir1
r_w_x  File       515  B  Wed Nov 29 09:30:24 2023   Wed Nov 29 10:53:43 2023   Wed Nov 29 10:53:39 2023   file1
r_w_x  File       0    B  Wed Nov 29 09:52:23 2023   Wed Nov 29 09:52:23 2023   Wed Nov 29 09:52:23 2023   file2
```

- 第一次输入 103 个 a 覆盖文件原有内容，再追加 103x4 个字符，总共 515 个字符，一个数据块存不下，需要在写入时分配数据块，可以从文件大小及读出结果看出成功分配了数据块。
- 从 ls 的结果可以看出，file1 的 modifytime 在写入文件时被修改，lastaccesstime 在读文件时被修改

➢ 测试文件的关闭

```
[root@ /]# close file1
File file1 is closed successfully!
[root@ /]# read file1
File file1 has not been opened!
```

➢ 测试文件的访问权限

```
[root@ /]# open file2
File file2 is opened successfully!
[root@ /]# chmod file2 wx
Please input the password: 666
[root@ /]# read file2
No permission to read file file2 !
[root@ /]# ls
mode    type        size      CreateTime                LastAccessTime              ModifyTime                  name
r_w_x   Directory   52   B    Wed Nov 29 09:18:09 2023    Wed Nov 29 10:55:08 2023    Wed Nov 29 10:36:25 2023    .
r_w_x   Directory   52   B    Wed Nov 29 09:18:09 2023    Wed Nov 29 10:55:08 2023    Wed Nov 29 10:36:25 2023    ..
r_w_x   Directory   42   B    Wed Nov 29 09:20:24 2023    Wed Nov 29 10:02:59 2023    Wed Nov 29 09:54:40 2023    dir1
r_w_x   File        103  B    Wed Nov 29 09:30:24 2023    Wed Nov 29 10:53:43 2023    Wed Nov 29 10:55:07 2023    file1
__w_x   File        0    B    Wed Nov 29 09:52:23 2023    Wed Nov 29 09:52:23 2023    Wed Nov 29 11:08:09 2023    file2
[root@ /]# chmod file2 rx
Please input the password: 666
[root@ /]# write file2 > a
No permission to write into file file2 !
[root@ /]# ls
mode    type        size      CreateTime                LastAccessTime              ModifyTime                  name
r_w_x   Directory   52   B    Wed Nov 29 09:18:09 2023    Wed Nov 29 11:08:22 2023    Wed Nov 29 10:36:25 2023    .
r_w_x   Directory   52   B    Wed Nov 29 09:18:09 2023    Wed Nov 29 11:08:22 2023    Wed Nov 29 10:36:25 2023    ..
r_w_x   Directory   42   B    Wed Nov 29 09:20:24 2023    Wed Nov 29 10:02:59 2023    Wed Nov 29 09:54:40 2023    dir1
r_w_x   File        103  B    Wed Nov 29 09:30:24 2023    Wed Nov 29 10:53:43 2023    Wed Nov 29 10:55:07 2023    file1
r___x   File        0    B    Wed Nov 29 09:52:23 2023    Wed Nov 29 09:52:23 2023    Wed Nov 29 11:08:43 2023    file2
```

(11) password：修改密码

```
[root@ /]# password
Please input the old password: 666
Please input the new password: xc2003
[root@ /]# chmod file2 rwx
Please input the password: 666
Wrong password!
[root@ /]# chmod file2 rwx
Please input the password: xc2003
```

- 修改访问权限时需要输入密码，这里使用这个功能来验证密码是否成功被修改

(12) login：登录　　logoff：登出

```
[root@ /]# login
Failed! You haven't logged out yet.
[root@ /]# logoff
Sure to log out? [Y/n]: y
[$$$]# login
Please input the password: xc2003
[root@ /]# ls
mode    type        size      CreateTime
r_w_x   Directory   52   B    Wed Nov 29 09:18:09 202
r_w_x   Directory   52   B    Wed Nov 29 09:18:09 202
r_w_x   Directory   42   B    Wed Nov 29 09:20:24 202
r_w_x   File        103  B    Wed Nov 29 09:30:24 202
r_w_x   File        0    B    Wed Nov 29 09:52:23 202
```

- 未登出时不能重复登录
- 登出后重新登录文件系统内容保持不变

(13) 若输入不存在的命令，系统会给出提示

```
[root@ /]# rmd
No this Command.Please check!
```

(14) 退出文件系统

```
[root@ /]# quit
Thank you for using! Bye~
[root@localhost expfile]#
```

(15) format:格式化

```
[root@ /]# ls
mode    type        size    CreateTime                  LastAccessTime              ModifyTime                  name
r_w_x   Directory   28  B   Wed Nov 29 15:21:33 2023    Wed Nov 29 15:21:33 2023    Wed Nov 29 15:21:45 2023    .
r_w_x   Directory   28  B   Wed Nov 29 15:21:33 2023    Wed Nov 29 15:21:33 2023    Wed Nov 29 15:21:45 2023    ..
r_w_x   Directory   17  B   Wed Nov 29 15:21:45 2023    Wed Nov 29 15:21:45 2023    Wed Nov 29 15:21:45 2023    dir1
[root@ /]# format
Sure to format the file system? [Y/n]: y
Creating the ext2 file system
The ext2 file system has been installed!
[root@ /]# ls
mode    type        size    CreateTime                  LastAccessTime              ModifyTime                  name
r_w_x   Directory   17  B   Wed Nov 29 15:22:04 2023    Wed Nov 29 15:22:04 2023    Wed Nov 29 15:22:04 2023    .
r_w_x   Directory   17  B   Wed Nov 29 15:22:04 2023    Wed Nov 29 15:22:04 2023    Wed Nov 29 15:22:04 2023    ..
```

(16) 测试多级索引

```
[root@ /]# mkf f1
File f1 is created successfully!
[root@ /]# open f1
File f1 is opened successfully!
[root@ /]# write f1 > aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

```
[root@ /]# write f1 >> aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
[root@ /]# ls
mode    type        size     CreateTime                  LastAccessTime              ModifyTime                  name
r_w_x   Directory   26   B   Wed Nov 29 15:27:06 2023    Wed Nov 29 15:27:06 2023    Wed Nov 29 15:27:18 2023    .
r_w_x   Directory   26   B   Wed Nov 29 15:27:06 2023    Wed Nov 29 15:27:06 2023    Wed Nov 29 15:27:18 2023    ..
r_w_x   File        5000 B   Wed Nov 29 15:27:18 2023    Wed Nov 29 15:27:18 2023    Wed Nov 29 15:28:05 2023    f1
[root@ /]# read f1

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

```
[root@ /]# rmf f1
File f1 is closed successfully!
f1 is deleted successfully!
[root@ /]# ls
mode    type        size     CreateTime                  LastAccessTime              ModifyTime                  name
r_w_x   Directory   17   B   Wed Nov 29 15:27:06 2023    Wed Nov 29 15:28:14 2023    Wed Nov 29 15:28:32 2023    .
r_w_x   Directory   17   B   Wed Nov 29 15:27:06 2023    Wed Nov 29 15:28:14 2023    Wed Nov 29 15:28:32 2023    ..
```

输入两次 2500 个字符，共 5000 个字符，需要 10 个数据块来存储（如果没有索引文件是无法分配这么多块的）。从运行结果可以看出，文件成功分配了 5000 字节空间并能读出 5000 个 a（太长了，没有放全），之后成功删除文件并在删除时关闭了文件。

1.6 实验总结

1.6.1 实验中的问题与解决方法

(1) 用来模拟磁盘的文件始终为空

原因：使用 fopen 打开文件时未正确设置权限，每次打开时都使用 wb+清空了文件

解决方法：通过查阅资料设置了正确的权限

https://blog.csdn.net/qq_37037348/article/details/130563950

(2) 输出的同一个结点的三个时间始终相同

原因：localtime 函数实现的时候采用的是一块固定 buffer，因此如果多次调用此函数，结果值会是最后一次的结果值

解决方法：在 localtime 函数调用后直接将 buffer 内容拷贝出来

https://blog.csdn.net/cleanfield/article/details/6224804

(3) 每次需要从磁盘中读出一整个块，如何从读出的数据块缓冲区中分离出需要的目录项结构体

解决方法：使用 memcpy 函数可以在不同类型的变量之间进行拷贝

https://blog.csdn.net/GoodLinGL/article/details/114602721?ops_request_misc=%257B%2522request%255Fid%2522%253A%252217012437161680022289986%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=170124371616800222899886&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-114602721-null-null.142^v96^pc_search_result_base3&utm_term=memcpy&spm=1018.2226.3001.4187

1.6.2 实验收获

(1) 理论知识的应用： 通过实验，我将之前学到的文件系统的理论知识应用到实际的代码编写中。这加深了我对文件系统工作原理的理解，特别是对 Ext2 文件系统的底层结构和算法。

(2) **模拟与实际硬件的关联：** 通过使用现有操作系统上的文件模拟硬盘，我更好地理解了文件系统是如何与实际硬件交互的。这种模拟方法让我能够在软件中模拟硬件层的行为。

(3) **多级索引的实现：** 实现多级索引是一个挑战，但它使我更深入地了解了文件系统中索引结构的复杂性。这也增加了我对文件存储和检索的实际实现的认识。

(4) **系统设计与代码组织：** 实验要求分为底层函数、命令层函数和用户接口层函数，这种模块化的设计让代码更具可读性和可维护性。这也强调了良好的系统设计在大型项目中的重要性。

(5) 错误处理和异常情况考虑： 在编写代码时，我意识到错误处理和异常情况的考虑是至关重要的。文件系统需要考虑并处理各种可能出现的问题。

(6) 系统的实用性： 实验中实现了一系列实际文件系统应该有的功能，如文件的增删改查、

权限控制等。这使得我对实际文件系统的工作更有直观的认识。

总体来说，这个实验是一个很好的学习体验，通过亲自实践文件系统的设计和实现，我更深入地理解了文件系统的内部机制，提高了我的编程能力和系统设计水平。

1.6.3 意见与建议

(1) 建议给出 linux 文件系统下对应每种操作的命令，让我们可以在实际的虚拟机或服务器中直接输入命令验证，更加深入理解这些文件操作的具体功能并模拟实现。

(2) 个人认为模拟文件系统中可以适当缩小数据块大小，这样便于验证多级索引的功能。