|  |
| --- |
| 杭州电子科技大学 |
| 简单文件系统的实现 |
| 操作系统实验 |

|  |
| --- |
| xxxxx  2019-6-12 |

目录

[一、 设计要求 4](#_Toc11398085)

[1. 设计目的 4](#_Toc11398086)

[2. 内容要求 4](#_Toc11398087)

[3. 学时安排（共12学时） 5](#_Toc11398088)

[4. 开发平台 5](#_Toc11398089)

[二、 设计方案 5](#_Toc11398090)

[1. 文件控制块 5](#_Toc11398091)

[2. 文件分配表 6](#_Toc11398092)

[3. 用户打开表 7](#_Toc11398093)

[4. 引导块 8](#_Toc11398094)

[5. 实验中主要函数接口设计 8](#_Toc11398095)

[6. 主要函数设计思想 9](#_Toc11398096)

[1) Main函数 9](#_Toc11398097)

[2) 文件系统函数:startsys() 9](#_Toc11398098)

[3) 磁盘格式化函数:my\_format() 10](#_Toc11398099)

[4) 更改当前目录函数:my\_cd() 11](#_Toc11398100)

[5) 创建子目录函数:my\_mkdir() 12](#_Toc11398101)

[6) 删除子目录函数:my\_rmdir() 13](#_Toc11398102)

[7) 显示目录函数:my\_ls() 13](#_Toc11398103)

[8) 创建文件函数:my\_create() 14](#_Toc11398104)

[9) 删除文件函数:my\_rm() 15](#_Toc11398105)

[10) 打开文件函数:my\_open() 16](#_Toc11398106)

[11) 关闭文件函数:my\_close() 17](#_Toc11398107)

[12) 写文件函数:my\_write() 17](#_Toc11398108)

[13) 实际写文件函数:do\_write() 19](#_Toc11398109)

[14) 读文件函数:my\_read() 20](#_Toc11398110)

[15) 实际文件函数:do\_read() 21](#_Toc11398111)

[16) 退出文件函数:my\_exitsys() 22](#_Toc11398112)

[三、 程序测试截图 23](#_Toc11398113)

[1. 创建和删除目录 23](#_Toc11398114)

[2. 创建文件并写入读出最后删除 24](#_Toc11398115)

[3. 退出之后再次打开仍可读取文件系统并使用 25](#_Toc11398116)

[4. 创新功能打印fat表 25](#_Toc11398117)

# 设计要求

## 设计目的

通过具体的文件存储空间的管理、文件的物理结构、目录结构和文件操作的实现，加深对文件系统内部数据结构、功能以及实现过程的理解。

## 内容要求

1. 在内存中开辟一个虚拟磁盘空间作为文件存储分区，在其上实现一个简单的基于多级目录的单用户单任务系统中的文件系统。在退出该文件系统的使用时，应将该虚拟文件系统以一个Windows 文件的方式保存到磁盘上，以便下次可以再将它恢复到内存的虚拟磁盘空间中。
2. 文件存储空间的分配可采用显式链接分配或其他的办法。
3. 空闲磁盘空间的管理可选择FAT表、位示图或其他办法。
4. 文件目录结构采用多级目录结构。为了简单起见，可以不使用索引结点，其中的每个目录项应包含文件名、物理地址、长度等信息，还可以通过目录项实现对文件的读和写的保护。
5. 要求提供以下操作命令：

my\_format：对文件存储器进行格式化，即按照文件系统的结构对虚拟磁盘空间进行布局，并在其上创建根目录以及用于管理文件存储空间等的数据结构。

my\_mkdir：用于创建子目录。

my\_rmdir：用于删除子目录。

my\_ls：用于显示目录中的内容。

my\_cd：用于更改当前目录。

my\_create：用于创建文件。

my\_open：用于打开文件。

my\_close：用于关闭文件。

my\_write：用于写文件。

my\_read：用于读文件。

my\_rm：用于删除文件。

my\_exitsys：用于退出文件系统。

## 学时安排（共12学时）

## 开发平台

Linux环境，gcc，gdb，vim或gedit等。

# 开发平台

Ubuntu，gcc，vim，vscode，gdb等。

# 设计实现方案

## 文件控制块

|  |
| --- |
| */\* 文件控制块 \*/*  typedef **struct** FCB  {  **char** filename[8];          *// 文件名*  **char** exname[3];            *// 文件扩展名*  **unsigned** **char** attribute;       *// 文件属性字段，值为0时表示目录文件，值为1时表示数据文件*  **unsigned** **short** time;       *// 文件创建时间*  **unsigned** **short** date;       *// 文件创建日期*  **unsigned** **short** first;      *// 文件起始盘块号*  **unsigned** **long** length;      *// 文件长度*  **char** free;             *// 表示目录项是否为空，若值为0，表示空，值为1，表示已分配*  }fcb; |

## 文件分配表

为了实现文件的链式存储,硬盘上必须准确地记录哪些簇已经被文件占用,

还必须为每个已经占用的簇指明存储后继内容的下一个簇的簇号,对一个文件的最后一簇,

则要指明本簇无后继簇。这些都是由FAT表来保存的,表中有很多表项,每项记录一个簇的信息

|  |
| --- |
| */\* 文件分配表 \*/*  */\*为了实现文件的链式存储,硬盘上必须准确地记录哪些簇已经被文件占用,*  *还必须为每个已经占用的簇指明存储后继内容的下一个簇的簇号,对一个文件的最后一簇,*  *则要指明本簇无后继簇。这些都是由FAT表来保存的,表中有很多表项,每项记录一个簇的信息。\*/*  typedef **struct** FAT  {  **unsigned** **short** id; *// 磁盘块的状态（空闲的，最后的，下一个）*  }fat; |

## 用户打开表

|  |
| --- |
| */\* 用户打开文件表 \*/*  typedef **struct** USEROPEN  {  **char** filename[8]; *// 文件名*  **char** exname[3]; *// 文件扩展名*  **unsigned** **char** attribute;*//文件属性字段，值为0时表示目录文件，值为1时表示数据文件*  **unsigned** **short** time; *// 文件创建时间*  **unsigned** **short** date; *// 文件创建日期*  **unsigned** **short** first; *// 文件起始盘块号*  **unsigned** **long** length;*//文件长度（对数据文件是字节数，对目录文件可以是目录项个数）*  **char** free; *// 表示目录项是否为空，若值为0，表示空，值为1，表示已分配*  **unsigned** **short** dirno;   *// 相应打开文件的目录项在父目录文件中的盘块号*  **int** diroff;            *// 相应打开文件的目录项在父目录文件的dirno盘块中的目录项序号*  **char** dir[80];          *// 相应打开文件所在的路径名，这样方便快速检查出指定文件是否已经打开*  **int** father;            *// 父目录在打开文件表项的位置*  **int** count;             *// 读写指针在文件中的位置,文件的总字符数*  **char** fcbstate;         *// 是否修改了文件的FCB的内容，如果修改了置为1，否则为0*  **char** topenfile;        *// 表示该用户打开表项是否为空，若值为0，表示为空，否则表示已被某打开文件占据*  }useropen; |

## 引导块

|  |
| --- |
| */\* 引导块 \*/*  typedef **struct** BLOCK0  {  **char** magic[10]; *// 文件系统魔数*  **char** information[200];*//存储一些描述信息，如磁盘块大小、磁盘块数量、最多打开文件数等*  **unsigned** **short** root; *// 根目录文件的起始盘块号*  **unsigned** **char** \*startblock; *// 虚拟磁盘上数据区开始位置*  }block0; |

## 实验中主要函数接口设计

|  |
| --- |
| **void** startsys();                               *// 进入文件系统*  **void** my\_format();                               *// 磁盘格式化*  **void** my\_cd(**char** \*dirname);                             *// 更改当前目录*  **void** my\_mkdir(**char** \*dirname);                          *// 创建子目录*  **void** my\_rmdir(**char** \*dirname);                          *// 删除子目录*  **void** my\_ls();                                  *// 显示目录*  **void** my\_create (**char** \*filename);                       *// 创建文件*  **void** my\_rm(**char** \*filename);                            *// 删除文件*  **int** my\_open(**char** \*filename);                           *// 打开文件*  **int** my\_close(**int** fd);                              *// 关闭文件*  **int** my\_write(**int** fd);                              *// 写文件*  **int** do\_write(**int** fd, **char** \*text, **int** len, **char** wstyle);             *// 实际写文件*  **int** my\_read (**int** fd, **int** len);                         *// 读文件*  **int** do\_read (**int** fd, **int** len,**char** \*text);                  *// 实际读文件*  **void** my\_fat();                                  *//显示fat表*  **void** my\_exitsys();                                 *// 退出文件系统*  **unsigned** **short** findblock();                            *// 寻找空闲盘块*  **int** findopenfile();                                *// 寻找空闲文件表项* |

## 主要函数设计思想

主要函数思想



### Main函数

① 对前面定义的全局变量进行初始化；

② 调用 startsys()进入文件系统；

③ 列出文件系统提供的各项功能及命令调用格式；

④ 显示命令行提示符，等待用户输入命令；

⑤ 将用户输入的命令保存到一个 buf 中；

⑥ 对 buf 中内容进行命令解析，并调用相应函数执行用户键入的命令；

⑦ 如果命令不是“my\_exitsys”，则命令执行完毕后转④。

### 文件系统函数:startsys()

① 申请虚拟磁盘空间；

② 使用 c 语言的库函数 fopen()打开 myfsys 文件：若文件存在，则转

③；若文件不存在，则创建之，转⑤

③ 使用 c 语言的库函数 fread()读入 myfsys 文件内容到用户空间中的一个缓冲区中，并判断其开始的 8 个字节内容是否为“10101010”（文件系统魔数），如果是，则转④；否则转⑤；

④ 将上述缓冲区中的内容复制到内存中的虚拟磁盘空间中；转⑦

⑤ 在屏幕上显示“myfsys 文件系统不存在，现在开始创建文件系统”信

息，并调用 my\_format()对①中申请到的虚拟磁盘空间进行格式化操作。

转⑥；

⑥ 将虚拟磁盘中的内容保存到 myfsys 文件中；转⑦

⑦ 使用 c 语言的库函数 fclose()关闭 myfsys 文件；

⑧ 初始化用户打开文件表，将表项 0 分配给根目录文件使用，并填写根目录文件的相关信息，由于根目录没有上级目录，所以表项中的 dirno 和diroff 分别置为 5（根目录所在起始块号）和 0；并将 ptrcurdir 指针指向该用户打开文件表项。

⑨ 将当前目录设置为根目录。

### 磁盘格式化函数:my\_format()

① 将虚拟磁盘第一个块作为引导块，开始的 8 个字节是文件系统的魔数，记为“10101010”；在之后写入文件系统的描述信息，如 FAT 表大小及位置、根目录大小及位置、盘块大小、盘块数量、数据区开始位置等信息；

② 在引导块后建立两张完全一样的 FAT 表，用于记录文件所占据的磁盘块及管理虚拟磁盘块的分配，每个 FAT 占据两个磁盘块；对于每个 FAT 中，前面 5 个块设置为已分配，后面 995 个块设置为空闲；

③ 在第二张 FAT 后创建根目录文件 root，将数据区的第 1 块（即虚拟磁盘的第 6 块）分配给根目录文件，在该磁盘上创建两个特殊的目录项：

“.”和“..”，其内容除了文件名不同之外，其他字段完全相同。

### 更改当前目录函数:my\_cd()

① 调用 my\_open()打开指定目录名的父目录文件，并调用 do\_read()读入该父目录文件内容到内存中；

② 在父目录文件中检查新的当前目录名是否存在，如果存在则转③，否则返回，并显示出错信息；

③ 调用 my\_close()关闭①中打开的父目录文件；

④ 调用 my\_close()关闭原当前目录文件；

⑤ 如果新的当前目录文件没有打开，则打开该目录文件；并将 ptrcurdir指向该打开文件表项；

⑥ 设置当前目录为该目录。

### 创建子目录函数:my\_mkdir()

① 调用 do\_read()读入当前目录文件内容到内存，检查当前目录下新建目录文件是否重名，若重名则返回，并显示错误信息；

② 为新建子目录文件分配一个空闲打开文件表项，如果没有空闲表项则返回-1，并显示错误信息；

③ 检查 FAT 是否有空闲的盘块，如有则为新建目录文件分配一个盘块，否则释放①中分配的打开文件表项，返回，并显示错误信息；

④ 在当前目录中为新建目录文件寻找一个空闲的目录项或为其追加一个新的目录项;需修改当前目录文件的长度信息，并将当前目录文件的用户打开文件表项中的 fcbstate 置为 1；

⑤ 准备好新建目录文件的 FCB 的内容，文件的属性为目录文件，以覆盖写方式调用 do\_write()将其填写到对应的空目录项中；

⑥ 在新建目录文件所分配到的磁盘块中建立两个特殊的目录项“.”和

“..”目录项，方法是：首先在用户空间中准备好内容，然后以截断写或

者覆盖写方式调用 do\_write()将其写到③中分配到的磁盘块中；

⑦ 返回。

### 删除子目录函数:my\_rmdir()

① 调用 do\_read()读入当前目录文件内容到内存，检查当前目录下欲删除

目录文件是否存在，若不存在则返回，并显示错误信息；

② 检查欲删除目录文件是否为空（除了“.”和“..”外没有其他子目录

和文件），可根据其目录项中记录的文件长度来判断，若不为空则返回，

并显示错误信息；

③ 检查该目录文件是否已经打开，若已打开则调用 my\_close()关闭掉；

④ 回收该目录文件所占据的磁盘块，修改 FAT；

⑤ 从当前目录文件中清空该目录文件的目录项，且 free 字段置为 0：以

覆盖写方式调用 do\_write()来实现；

⑥ 修改当前目录文件的用户打开表项中的长度信息，并将表项中的fcbstate 置为 1；

⑦ 返回。

### 显示目录函数:my\_ls()

① 调用 do\_read()读出当前目录文件内容到内存；

② 将读出的目录文件的信息按照一定的格式显示到屏幕上；

③ 返回。

### 创建文件函数:my\_create()

① 为新文件分配一个空闲打开文件表项，如果没有空闲表项则返回-1，并显示错误信息；

② 若新文件的父目录文件还没有打开，则调用 my\_open()打开；若打开失败，则释放①中为新建文件分配的空闲文件打开表项，返回-1，并显示错误信息；

③ 调用 do\_read()读出该父目录文件内容到内存，检查该目录下新文件是否重名，若重名则释放①中分配的打开文件表项，并调用 my\_close()关闭

②中打开的目录文件；然后返回-1，并显示错误信息；

④ 检查 FAT 是否有空闲的盘块，如有则为新文件分配一个盘块，否则释放①中分配的打开文件表项，并调用 my\_close()关闭②中打开的目录文件；返回-1，并显示错误信息；

⑤ 在父目录中为新文件寻找一个空闲的目录项或为其追加一个新的目录项;需修改该目录文件的长度信息，并将该目录文件的用户打开文件表项中的 fcbstate 置为 1；

⑥ 准备好新文件的 FCB 的内容，文件的属性为数据文件，长度为 0，以覆

盖写方式调用 do\_write()将其填写到⑤中分配到的空目录项中；

⑦ 为新文件填写①中分配到的空闲打开文件表项，fcbstate 字段值为 0，读写指针值为 0；

⑧ 调用 my\_close()关闭②中打开的父目录文件；

⑨ 将新文件的打开文件表项序号作为其文件描述符返回。

### 删除文件函数:my\_rm()

① 若欲删除文件的父目录文件还没有打开，则调用 my\_open()打开；若打开失败，则返回，并显示错误信息；

② 调用 do\_read()读出该父目录文件内容到内存，检查该目录下欲删除文件是否存在，若不存在则返回，并显示错误信息；

③ 检查该文件是否已经打开，若已打开则关闭掉；

④ 回收该文件所占据的磁盘块，修改 FAT；

⑤ 从文件的父目录文件中清空该文件的目录项，且 free 字段置为 0：以覆盖写方式调用 do\_write()来实现；；

⑥ 修改该父目录文件的用户打开文件表项中的长度信息，并将该表项中的fcbstate 置为 1；

⑦ 返回。

### 打开文件函数:my\_open()

① 检查该文件是否已经打开，若已打开则返回-1，并显示错误信息；

② 调用 do\_read()读出父目录文件的内容到内存，检查该目录下欲打开文件是否存在，若不存在则返回-1，并显示错误信息；

③ 检查用户打开文件表中是否有空表项，若有则为欲打开文件分配一个空表项，若没有则返回-1，并显示错误信息；

④ 为该文件填写空白用户打开文件表表项内容，读写指针置为 0；

⑤ 将该文件所分配到的空白用户打开文件表表项序号（数组下标）作为文件描述关闭文件函数:my\_close()符 fd 返回。

### 关闭文件函数:my\_close()

① 检查 fd 的有效性（fd 不能超出用户打开文件表所在数组的最大下标），如果无效则返回-1；

② 检查用户打开文件表表项中的 fcbstate 字段的值，如果为 1 则需要将该文件的 FCB 的内容保存到虚拟磁盘上该文件的目录项中，方法是：打开该文件的父目录文件，以覆盖写方式调用 do\_write()将欲关闭文件的 FCB写入父目录文件的相应盘块中；

③ 回收该文件占据的用户打开文件表表项（进行清空操作），并将

topenfile 字段置为 0；

④ 返回。

### 写文件函数:my\_write()

① 检查 fd 的有效性（fd 不能超出用户打开文件表所在数组的最大下

标），无效则返回-1，并显示出错信息；

② 提示并等待用户输入写方式：（1：截断写；2：覆盖写；3：追加

写）

③ 如果用户要求的写方式是截断写，则释放文件除第一块外的其他磁盘空间内容（查找并修改 FAT 表），将内存用户打开文件表项中文件长度修改

为 0，将读写指针置为 0 并转④；如果用户要求的写方式是追加写，则修改文件的当前读写指针位置到文件的末尾，并转④；如果写方式是覆盖写，则直接转④；

④ 提示用户：整个输入内容通过 CTR+Z 键（或其他设定的键）结束；用户可分多次输入写入内容，每次用回车结束；

⑤ 等待用户从键盘输入文件内容，并将用户的本次输入内容保存到一临时变量 text[]中，要求每次输入以回车结束，全部结束用 CTR+Z 键（或其他设定的键）；

⑥ 调用 do\_write()函数将通过键盘键入的内容写到文件中。

⑦ 如果 do\_write()函数的返回值为非负值，则将实际写入字节数增加do\_write()函数返回值，否则显示出错信息，并转⑨；

⑧ 如果 text[]中最后一个字符不是结束字符 CTR+Z，则转⑦继续进行写操作；否则转⑨；

⑨ 如果当前读写指针位置大于用户打开文件表项中的文件长度，则修改打开文件表项中的文件长度信息，并将 fcbstate 置 1；

⑩ 返回实际写入的字节数。

### 实际写文件函数:do\_write()

① 用 malloc()申请 1024B 的内存空间作为读写磁盘的缓冲区 buf，申请失

败则返回-1，并显示出错信息；

② 将读写指针转化为逻辑块块号和块内偏移 off，并利用打开文件表表项中的首块号及 FAT 表的相关内容将逻辑块块号转换成对应的磁盘块块号blkno；如果找不到对应的磁盘块，则需要检索 FAT 为该逻辑块分配一新的磁盘块，并将对应的磁盘块块号 blkno 登记到 FAT 中，若分配失败，则返回-1，并显示出错信息；

③ 如果是覆盖写，或者如果当前读写指针所对应的块内偏移 off 不等于0，则将块号为 blkno 的虚拟磁盘块全部 1024B 的内容读到缓冲区 buf 中；

否则便用 ASCII 码 0 清空 buf；

④ 将 text 中未写入的内容暂存到缓冲区 buff 的第 off 字节开始的位置，

直到缓冲区满，或者接收到结束字符 CTR+Z 为止；将本次写入字节数记录

到 tmplen 中；

⑤ 将 buf 中 1024B 的内容写入到块号为 blkno 的虚拟磁盘块中；

⑥将当前读写指针修改为原来的值加上 tmplen；并将本次实际写入的字节数增加 tmplen；

⑦ 如果 tmplen 小于 len，则转②继续写入；否则转⑧；

⑧ 返回本次实际写入的字节数。

### 读文件函数:my\_read()

① 定义一个字符型数组 text[len]，用来接收用户从文件中读出的文件内容；

② 检查 fd 的有效性（fd 不能超出用户打开文件表所在数组的最大下标），如果无效则返回-1，并显示出错信息；

③ 调用 do\_read()将指定文件中的 len 字节内容读出到 text[]中；

④ 如果 do\_read()的返回值为负，则显示出错信息；否则将 text[]中的内容显示到屏幕上；

⑤ 返回。

### 实际文件函数:do\_read()

① 使用 malloc()申请 1024B 空间作为缓冲区 buf，申请失败则返回-1，并显示出错信息；

② 将读写指针转化为逻辑块块号及块内偏移量 off，利用打开文件表表项中的首块号查找 FAT 表，找到该逻辑块所在的磁盘块块号；将该磁盘块块号转化为虚拟磁盘上的内存位置；

③ 将该内存位置开始的 1024B（一个磁盘块）内容读入 buf 中；

④ 比较 buf 中从偏移量 off 开始的剩余字节数是否大于等于应读写的字节数 len，如果是，则将从 off 开始的 buf 中的 len 长度的内容读入到text[]中；否则，将从 off 开始的 buf 中的剩余内容读入到 text[]中；

⑤ 将读写指针增加④中已读字节数，将应读写的字节数 len 减去④中已读字节数，若 len 大于 0，则转②；否则转⑥；

⑥ 使用 free()释放①中申请的 buf。

⑦ 返回实际读出的字节数。

### 退出文件函数:my\_exitsys()

① 使用 C 库函数 fopen()打开磁盘上的 myfsys 文件；

② 将虚拟磁盘空间中的所有内容保存到磁盘上的 myfsys 文件中；

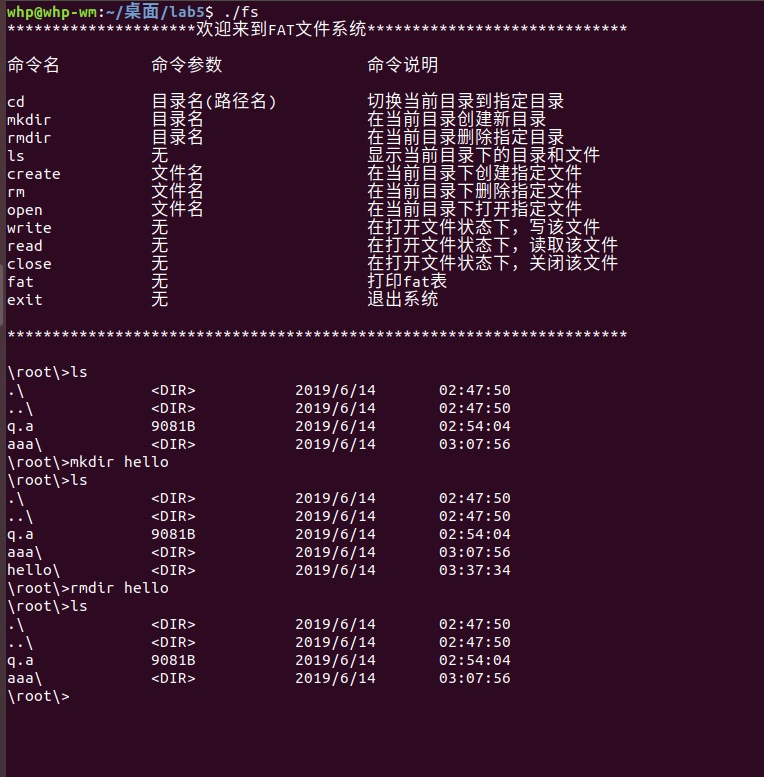
③ 使用 c 语言的库函数 fclose()关闭 myfsys 文件；

④ 撤销用户打开文件表，释放其内存空间

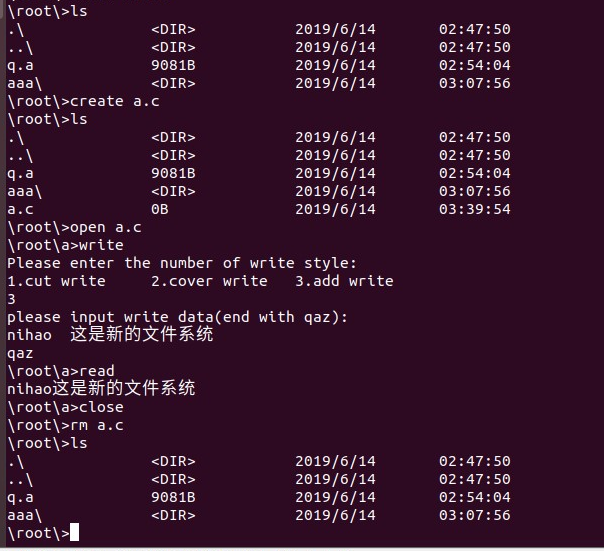
④ 释放虚拟磁盘空间。

# 程序测试截图

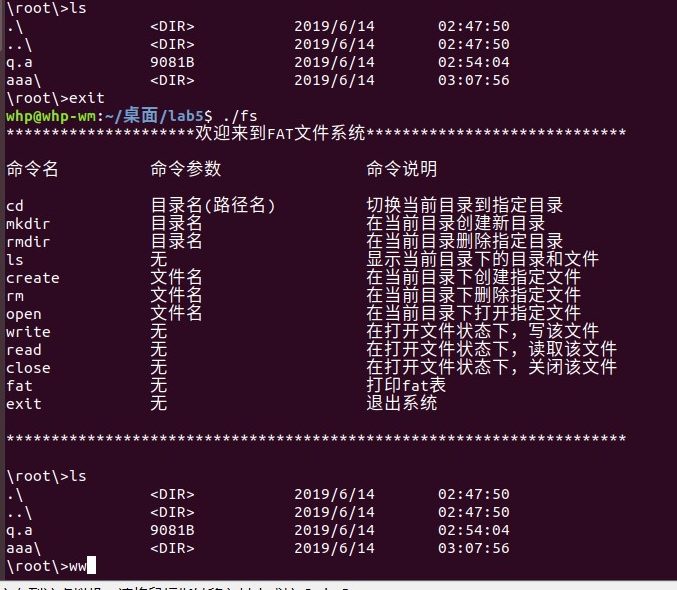
## 创建和删除目录



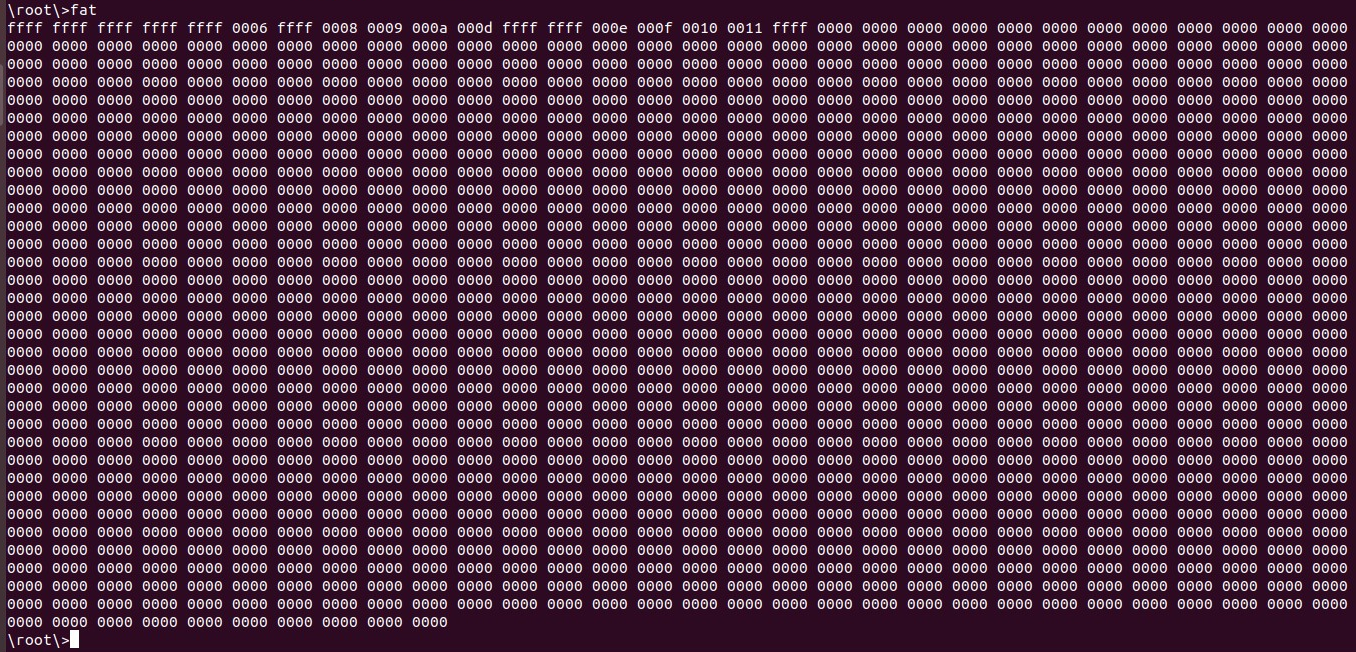
## 创建文件并写入读出最后删除



## 退出之后再次打开仍可读取文件系统并使用



## 创新功能打印fat表



# 实验遇到的问题及解决方式

## 追加写时，写入的内容没有保存到文件里

找此文件跨越的最后一个磁盘块时出错。文件开始指针应为数据区开始地址+文件打开表当前表项的 first\*BLOCKSIZE。

## 删除文件时问题

把当前目录第一个盘块号赋值给 fat 表的 id,通过 fat 表依次查找文件的下一个盘块，把 fat 的 id 设置为 free。若只将 fat 表中文件的表项释放，并没有将实际文件中的内容删除，防止数据被恢复，保障用户的数据安全。

根据当前目录的第一个盘块号的 id,磁盘块大小 BLOCKSIZE,得到id\*BLOCKSIZE 为文件的第一个磁盘块地址，通过循环找到文件的每个磁盘块并把所占的磁盘空间清空。

# 实验结果讨论

## 优点

程序实现创新性代码量少，代码可读性高。可以打印fat表直观

## 程序缺点

(1) 删除目录文件时，若欲删除目录文件为空，正常删除该目录文件；不为空则直接返回，并显示错误信息。可以改进为若不为空，提示用户是否将目录内文件同时删除，若用户同意，通过 FCB 表找到该目录下的子文件，先将子文件删除，再删除整个父目录文件。

(2) 文件系统内文件的名称不支持中文以及其他 unicode 码，不支持 emoji 表情作为文件名。

(3) 程序崩溃时数据无法保存。

## 程序考虑的改进思路

(1) 增加函数模块，增加调用命令。

(2) 增加多种编码格式。

(3) 增加一个账号登录，提高安全保障。