

操作系统课程设计

1552729 李依璇
1652690 苏昭帆

同济大学

目录

一、 团队介绍.....	3
二、 开发环境及安装说明.....	3
1. 开发环境.....	3
2. 开发语言.....	3
3. 安装说明.....	3
三、 操作系统功能设计.....	3
1. 总体功能.....	3
2. 操作系统组成.....	3
3. 具体功能介绍.....	3
四、 源码中遇到的 BUG.....	5
a. 光盘源码 make 报错.....	5
b. Alt+Fn 无法切换 TTY.....	5
c. GCC 编译不通过.....	5
d. 解压 cmd.tar 时在 pwd 程序使用 write() 写进硬盘时卡住.....	5
五、 修改的源码.....	5
六、 操作系统使用说明.....	6

一、团队介绍

1552729 李依璇：应用制作、系统时间、TTY 操作、开机动画、文档、PPT 50%

1652690 苏昭帆：多级文件系统及持久化制作、底层代码修改、部分开机动画、文档、PPT 50%

二、开发环境及安装说明

1. 开发环境

- ubuntu/kylin-16.04-32bit
- bochs-2.6.9

2. 开发语言

C 语言、汇编语言

3. 安装说明

在终端进入项目文件之后，依次输入以下命令：

```
$make image  
$cd ./command  
$make install  
$cd ..  
$bochs
```

三、操作系统功能设计

1. 总体功能

实现了微内核进程间通信，多级文件系统，内存管理并最终在当前终端上显示一个 shell 的操作系统。

2. 操作系统组成

boot（引导）、command（应用集）
fs（文件系统）、include（头文件集）
kernel（内核）、lib（可用代码库）
mm（内存调度系统）

3. 具体功能介绍

a. 多级文件系统

对 Orange fs 文件夹下的源码进行修改,使得能够直接向源码中文件操作接口传入多级目录后就能对指定的正确的文件进行操作,而不需要在上层使用一个文件存储目录、对目录进行解析。也就是说,在 shell 中向 open() 函数直接传入用户请求的多级目录,就能够返回正确的文件描述符 fd。

本项目多级文件系统的底层实现中,文件与原来相同,文件夹结构与原来扁平系统中的根目录相同、大小与普通文件大小相同。

为实现该多级文件系统,在代码中进行了下述修改:

修改 fs/link.c: 增加了 deletedirectory() 函数,以实现对文件夹的删除;增加 do_unlink_my() 函数,使得能够根据文件类型 (type=0, 为文件) 进行文件链接或者文件夹链接的删除;增加了 recursivedelete() 函数,进行递归删除,当索引结点是文件时,直接删除文件,当索引结点是文件夹时则递归删除文件夹。

修改 fs/misc.c: 不再使用原来的 strip_path() 函数,将其改为 strip_path_dir() 函数,能够通过全路径进行文件查找,是分离多级目录结构的核心。

修改 fs/open.c: 增加函数 new_inode_dir()、create_directory(), 实现增加新目录、设置目录结点为索引结点的功能,并在创建文件夹之后,能在磁盘上为该文件夹分配位置;增加 tempsearch() 函数,能够根据路径查找相应的文件的名称、父文件以及根;增加 printallchildfiles(), 来判断是否打印某一文件夹下的全部文件;更改原来的 do_open() 函数,使得当文件不存在于磁盘上时,创建一个新的文件或者文件夹。

修改 lib/open.c: 增加一个参数,通过消息传递机制将是文件还是文件夹的信息传递到 fs。

其他小的修改: 比如消息中需要多添加一项 FILETYPE (文件 or 文件夹) 等。

最后,本项目参照 orange 第 11 章的内容进行了持久化实现,在 fs/main.c 中的 init_fs() 函数初始化文件系统时,每次先读取超级块,如果发现了 magic number,则认为分区已经装上了文件系统,否则调用 mkfs(), 制作一个文件系统。同时在 kernel/main.c 中的 untar() 函数中增加一个记号,当看到该记号后,操作系统不再对 cmd.tar 进行解包。如果解包时文件已存在于磁盘上,则需将原来的文件内容删除,写入新内容,故引入 O_TRUNC, 加入到 open() 的参数中,同时需要修改 fs 中的 do_open() 函数。

b. 多功能控制台

本项目中的控制台可以完成文件操作,使用常规应用和游戏;进行该控制台的实现需要对 kernel/main.c 中的 shabby_shell 进行修改。控制台中输入的命令分为两类,一类去读取已经安装在硬盘上并读入文件系统的应用 (即游戏),如果读取成功则执行该游戏,如果读取失败,即不存在这样的游戏,则会比对系统内部指令,比对成功则执行相应的系统内部指令 (例如 ls, cd 等)。

c. 游戏及应用

• 五子棋控制台游戏

使用评估函数算法实现人机对弈,1 为用户,2 为机器,对战棋盘为 8*8 布局,可以随时退出游戏。

• 扫雷控制台游戏

游戏布局为 9*9 的棋盘,棋盘上共有 10 个雷,用户只有在点开除了雷之

外的所有位置才能获得胜利，可以随时退出游戏。

- **计算机控制台应用**

可以进行包括正数和负数在内的二元整数的四则运算，可以随时退出应用。

d. 读取系统时间

利用消息传递机制实现一个系统调用来获取时间，用户接口是 kernel/main.c 中的 get_time 函数，消息类型为 orange 自带的 GET_RTC_TIME。处理函数在 kernel/systask.c 中的 get_rtc_time 函数中，用以读取操作系统的时间。

e. 进程管理

能够展示当前正在运行的进程的具体信息，分别展示系统任务和用户进程。

f. 多 TTY 操作

通过使用 Ctrl+Fn 组合键来解决在 bochs 中无法使用 Alt+Fn 组合键进行控制台切换的问题，从而实现了 TTY 的切换。

四、源码中遇到的 BUG

a. 光盘源码 make 报错

原因：ubuntu 18.04 LTS 64 位，书中默认是在 32 位环境下开发的

解决：ubuntu 18.04 不支持 32 位，固改用 ubutun 16.04 LTS 32 位进行源码编译

b. Alt+Fn 无法切换 TTY

原因：bochs 下使用 Alt+Fn 组合键只会响应 Ubuntu 的全局快捷键，虚拟机中的操作无法响应。

解决：将 tty.c 文件中的 Alt+Fn 组合键改为 Ctrl+Fn 组合键

c. GCC 编译不通过

原因：编译器版本问题

解决：在 MakeFile 中的 CFLAGS 加上 “ -fno-stack-protector ”

d. 解压 cmd.tar 时在 pwd 程序使用 write() 写进硬盘时卡住

原因：并行代码共享数据不同步

解决：在 kernel/proc.c 的

第 468 行添加 disable_in(p_who_wanna_recv->p_flage),

第 480 行添加 enable_in(p_who_wanna_recv->p_flags)

五、修改的源码

a. 将 fs/main.c 中的 strip_path(char* filename, const char* pathname, struct inode** ppinode) 函数更新为 strip_path_dir(char* filename, const char* pathname, struct inode** ppinode)。

函数作用：

1) pathname 为输入参数，filename 以及 ppinode 均为输出参数。该函数输入一个路径之后，能够得到最终要操作的文件/文件夹名称，存储在 filename 中；ppinode 则是指向 filename 上一级目录 inode 的指针。以输入 pathname 为 /a/m/n 为例，使用 strip_path_dir 函数之后，filename 为 m，ppinode 为一个指向 n 的 inode 指针。

2) 源码中的 strip_path 函数只能将 ppinode 置为 root_node，因为源码中的文件系统是一个扁平文件系统。

b. 在 fs/open.c 中新增 printallchildfiles(struct inode* pin) 函数

函数作用：

输出 pin 对应的文件夹下所有的内容（文件及文件夹）

c. 对 fs/open.c 中的 do_open 函数进行修改

函数作用：

1) 增加操作，使得根目录能够被 ls。如此一来，如果在 /a/ 下使用 ls，则会转化为 open 路径为 /a 的目录文件。

2) open 完成后再返回 fd 之前调用函数 printallchildfiles(pin) 函数并进行说明。pin 为要打开的目录/文件的 inode 指针。

3) 添加对 filetype 的判断，如果为创建，则根据 filetype 的不同调用 createfile 或者 createdirectory 函数并进行说明。

d. 在 fs/open.c 中新增 createdirectory 函数

函数作用：

实现文件夹的创建，创建其他分配位都与 createfile 函数相同，创建 inode 不同，该函数中，I_mode 为 I_DIRECTORY。

e. 将 fs/link.c 中的 do_unlink() 函数替换为 do_unlink_my() 函数

函数作用：

1) 实现删除文件以及递归删除文件夹。

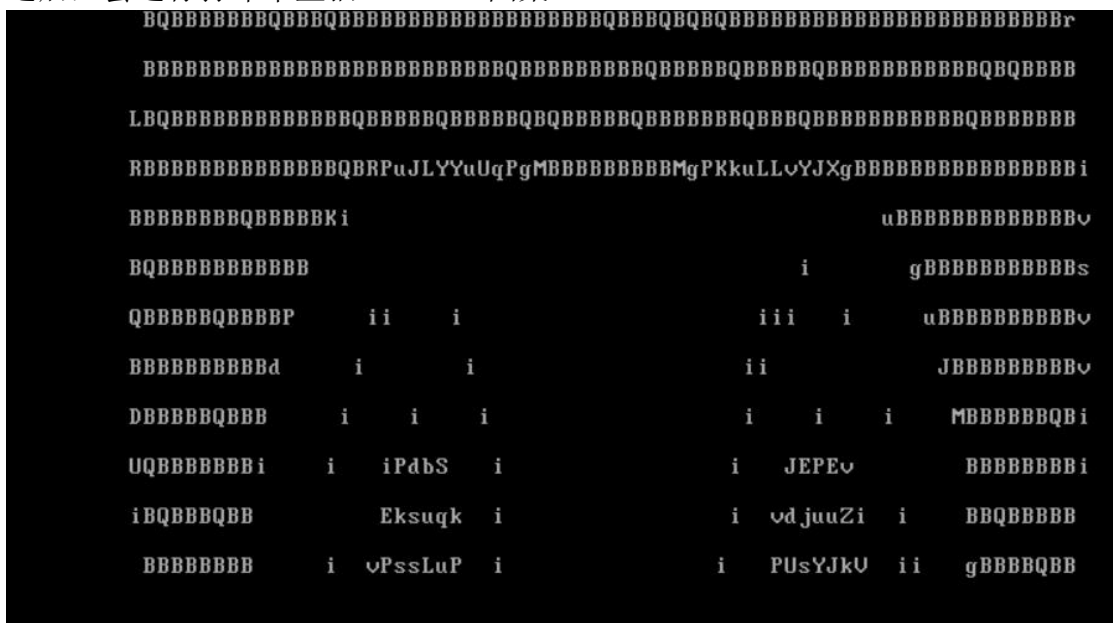
2) 实现目录存在的情况下，I_CNT 记录打开文件数依旧有效。

六、操作系统使用说明

进入操作系统之后首先展示欢迎界面，屏幕上出现一个彩色的 HI!!!。



之后，会逐行打印章鱼猫 Octocat 图案。




```

*****
help                |          List all commands
welcome             |          Welcome the users
octocat             |          Play the video
ls                  |          List all files in current file path
cd [dir]            |          Go into the dir
vim [file] [str]    |          Create a file
mkdir [dir]         |          Create a directory
cat [file]          |          Read a file
rm-f [file]         |          Delete a file
rm-d [dir]          |          Delete a directory
change [file][str]  |          Clear file content and write
vi-A [file][str]    |          Append content to a file
ps                  |          List process information
saolei              |          Start game saolei
fiveChess           |          Start game fiveChess
calculator          |          Start a binary caculator
time                |          Show current time
cls                 |          Clear the screen
*****
[root@virtual_machine]/$

```

指令: welcome

功能: 展示刚进入操作系统的欢迎界面，显示彩色的“HI!!!”

指令: octocat

功能: 展示章鱼猫动画

指令: ls

功能: 显示当前目录下文件和目录

```

[root@virtual_machine]/$ ls
dir:..
file:cmd.tar
file:kernel.bin
file:echo
file:pwd
file:calculator
file:saolei
file:calendar
file:fiveChess
dir:lyx
dir:szf

```

指令: cd [dir]

功能: 进入当前目录下名为 dir 的文件夹。cd .. 指令为返回上一级目录

指令 cd .. 返回上一级目录以及指令 cd szf 进入目录 szf 的运行结果如下:

```

[root@virtual_machine]/szf/$ cd ..
now location is /
[root@virtual_machine]/$ cd szf
location: /szf/

```

指令: vim [file] [str]

功能: 在当前目录下创建一个名为 file 的文件，并写入内容 str

指令 vim sfile1 test, 创建一个名为 sfile1 的文件，文件内容为 test。成功运行结果如下，显示文件内容为 test:

```
[root@virtual_machine1/szf/$ vim sfile1 test
file name: /szf/sfile1 content: test
content test
```

指令: mkdir [dir]

功能: 创建一个名为 dir 的目录（文件夹）

指令 mkdir szf, 创建一个名为 szf 的目录。运行结果如下:

```
[root@virtual_machine1/$ mkdir szf
create dir okdirectory name: /szf
```

指令: cat [file]

功能: 在当前目录下查看名为 file 的文件内容

指令 cat sfile1 查看 szf 目录下名为 sfile1 的文件的内容。成功运行结果如下, 显示文件中的内容 test:

```
[root@virtual_machine1/szf/$ cat sfile1
/szf/sfile1(fd=2) : test
```

指令: rm-f [file]

功能: 在当前目录下删除名为 file 的文件。若 file 不存在, 则显示没有这个文件, 若 file 存在, 则删除成功之后显示相应的提示信息。

指令 rm-f szfile1 删除 szf 目录下, 名为 szfile1 的文件以及指令 rm-f sfile1 删除 szf 目录下, 名为 sfile1 的文件。运行结果如下, 显示/szf/sfile1 已经被删除:

```
[root@virtual_machine1/szf/$ ls
file:sfile1
[root@virtual_machine1/szf/$ rm-f szfile1
{FS} FS::do_unlink(): search_file() returns invalid inode: /szf/szfile1
file removed fail: /szf/szfile1
[root@virtual_machine1/szf/$ rm-f sfile1
file removed: /szf/sfile1
```

指令: rm-d [dir]

功能: 删除名为 dir 的目录（文件夹）。若 dir 不存在, 则显示没有这个目录, 若 dir 存在, 则删除成功之后显示相应的提示信息。

指令 rm-d lyx 删除名为 lyx 的目录以及再次执行该指令的运行结果如下:

```
[root@virtual_machine1/$ rm-d lyx
directory removed: /lyx
[root@virtual_machine1/$ rm-d lyx
{FS} FS::do_unlink(): search_file() returns invalid inode: /lyx
directory removed fail: /lyx
```

指令: change [file][str]

功能: 在当前目录下将名为 file 的文件中的内容全部替换为 str 中的内容。

指令 change sfile1 test~~~将 szf 目录下名为 sfile1 的文件中的原内容 test 替换为 test~~~; 再执行 cat sfile1 指令, 读取 sfile1 文件中现在的内容。运行结果如下:

```
[root@virtual_machine1/szf/$ change sfile1 test~~~
edit succeed
[root@virtual_machine1/szf/$ cat sfile1
/szf/sfile1(fd=2) : test~~~
```

指令: vi-A [file][str]

功能: 在当前目录下以追加模式在名为 file 的文件中写入 str 中的内容

先执行指令 cat abc 读取文件 abc 中的内容, 为 hhh; 再执行指令 vi-A abc !!!, 在文件 abc 后面增加 “!!!” 内容; 最后执行指令 cat abc, 再次读取文件 abc 中的内容。运行结果如下:

```
[root@virtual_machine1/$ cat abc
/abc(fd=3) : hhh
[root@virtual_machine1/$ vi-A abc !!!
edit succeed
[root@virtual_machine1/$ cat abc
/abc(fd=4) : hhh!!!
```

指令: ps

功能: 展示当前包括用户进程以及系统进程在内的所有进程

```
[root@virtual_machine1/$ ps
***** System Process *****
pid: 0   Name: TTY      Priority: 15
pid: 1   Name: SYS      Priority: 15
pid: 2   Name: HD       Priority: 15
pid: 3   Name: FS       Priority: 15
pid: 4   Name: MM       Priority: 15

***** User Process *****
pid: 5   Name: INIT     Priority: 5   Running Status: 4
pid: 6   Name: TestA    Priority: 5   Running Status: 0
pid: 7   Name: TestB    Priority: 5   Running Status: 0
pid: 8   Name: TestC    Priority: 5   Running Status: 0
```

指令: saolei

功能: 开始扫雷游戏, 输入行与列进行排雷, 当成功选择除地雷之外的所有位置之后, 游戏成功。用户也可随时退出游戏。

游戏开始界面:

```
[root@virtual_machine1/$ saolei
*****
*          saolei          *
*****
*      1. 10 bombs total      *
*      2. Enter 1-9 row number *
*      3. Enter 1-9 col number *
*      4. Enter q to quit     *
*****

 1  2  3  4  5  6  7  8  9
1 * * * * * * * * *
2 * * * * * * * * *
3 * * * * * * * * *
4 * * * * * * * * *
5 * * * * * * * * *
6 * * * * * * * * *
7 * * * * * * * * *
8 * * * * * * * * *
9 * * * * * * * * *
Please input row number:
```

游戏结束界面:

```

1 * * * * *
2 * * * * *
3 * * * * *
4 * * * * *
5 * * * * *
6 * * * * *
7 * * * * *
8 * * * * *
9 * * * * *
Please input row number: 1
Please input col number: 3
BOOM SHAKALAKA! Game Over!

  1  2  3  4  5  6  7  8  9
1 0  0  1  0  0  0  0  0  0
2 0  0  0  0  0  1  0  0  0
3 0  0  0  0  0  1  0  0  0
4 1  0  0  0  0  0  0  0  0
5 0  0  0  1  0  0  0  0  0
6 0  0  0  0  0  0  0  1  0
7 0  0  1  0  0  0  0  0  1
8 0  0  0  0  0  0  0  1  0
9 0  0  0  0  1  0  0  0  0

[root@virtual_machine1]#

```

指令: fivechess

功能: 开始五子棋游戏，输入行列坐标进行下棋，为人机对弈模式，任意一方棋子 5 个相连时，该方获胜。

```

[root@virtual_machine1]# fiveChess
*****
*                    Five Chess                    *
*****
* 1. 1 refers to user, 2 refers to computer        *
* 2. - refers to available place                    *
* 3. Enter row number 0-7                          *
* 4. Enter col number 0-7                          *
* 5. Five chess in one line then you win            *
* 6. Enter q to quit                                *
*****

  0 1 2 3 4 5 6 7
-----
0 : - - - - -
1 : - - - - -
2 : - - - - -
3 : - - - - -
4 : - - - - -
5 : - - - - -
6 : - - - - -
7 : - - - - -

Please enter row number 0-7:

```

```

You:
  0 1 2 3 4 5 6 7
  -----
0 : - - - - -
1 : - - - - -
2 : - - 2 2 2 - -
3 : - 2 2 1 - -
4 : - 1 1 1 1 1 -
5 : - - - 1 - 2 -
6 : - - - - -
7 : - - - - -

Computer:
  0 1 2 3 4 5 6 7
  -----
0 : - - - - -
1 : - - - - -
2 : - - 2 2 2 - -
3 : - 2 2 1 - -
4 : 2 1 1 1 1 1 -
5 : - - - 1 - 2 -
6 : - - - - -
7 : - - - - -

Wow! You Win! ^_^

```

指令: calculator

功能: 开启计算器, 可以计算包括正数与负数在内的二元整数的四则运算
计算器开启界面:

```

[root@virtual_machine1]$ calculator
*****
*                      Calculator                      *
*****
*  1. Only two parameters                               *
*  2. Only for integer                                 *
*  3. Both positive and negative number                *
*  4. Enter q to quit                                  *
*****

Please input num1:

```

计算器使用界面:

```

Please input num1:1
num1: 1
Please input num2:3
num2: 3
Please input op( + - * / ):-
1 - 3 = -2

Please input num1:1
num1: 1
Please input num2:0
num2: 0
Please input op( + - * / ):/
Error!

Please input num1:
num1: 0
Please input num2:3
num2: 3
Please input op( + - * / ):/
0 / 3 = 0

Please input num1:q
[root@virtual_machine1]$

```

指令: time

功能: 获取当前系统时间


```

[root@virtual_machine1]$ time
2018/9/4 4:43:55

```

指令: `cls`
功能: 清屏

```
[root@virtual_machine1]# cls
```



```
[root@virtual_machine1]#
```

最后展示修复的源码中的一个 BUG，使得该操作系统可以支持多 TTY

```
[TTY #2]
```

