

操作系统原理实验

实验八

多终端、文件读写与操作

姓名： 吴侃

学号： 14348134

班别： 2014 级计算机系一班

日期： 2016.05.06 – 2016.05.19

目录

零、特色先览.....	3
一、实验目的.....	3
二、实验要求.....	3
三、实验环境与工具.....	3
(一) 实验环境.....	3
(二) 实验工具.....	4
四、实验方案.....	4
(1) 多终端实现.....	4
(2) FAT12 文件系统的文件读写.....	6
(3) FAT12 文件系统的文件操作.....	7
五、 实验操作.....	7
多终端:	7
文件读写:	10
测试 seekg, seekp.....	11
测试文件操作.....	12
六、 小结.....	13

零、特色先览

本次实验的亮点包括:

1. 实现了多用户 (终端)
2. 实现了文件读写 (open, write, read, seekp, seekg)
3. 实现了文件操作 (rm,cp,mv)

一、实验目的

实现多终端与 FAT12 文件系统中的文件读写、操作

二、实验要求

1. 实现多终端
2. 实现 FAT12 文件系统中的文件读写
3. 实现 FAT12 文件系统中的文件操作

三、实验环境与工具

(一) 实验环境

物理机操作系统: Arch Linux 4.5.4-1

调试使用虚拟机: qemu-system-i386, bochs

虚拟机软件: VMware Workstation 12 Pro

虚拟机配置: CPU: i7-4702MQ @ 2.20GHz, 使用单核单线程

内存:4 MB

硬盘:32 MB

(二) 实验工具

编辑器: Vim 7.4

汇编工具: NASM 2.11.08

C++编译器: g++ 6.1.1

链接工具: GNU ld 2.26.0.20160501

构建工具: GNU Make 4.1

调试工具: Bochs x86 Emulator 2.6.8

虚拟机: qemu-system-i386

VMWare Workstation 12 Pro

合并文件: dd

四、实验方案

(1) 多终端实现

多终端的实现，需要修改进程调度过程，将 shell 分离出内核，保存和恢复用户的屏幕信息与光标位置。

A. 进程调度

在 PCB 表中新增一项 UID，用来标识该进程所属用户的 ID，用 UserID 标记当前环境下的用户 ID。当 UID 等于 0 时，说明这是内核进程，该进程可以在任何用户环境下运行。其他用户进程，只在进程的 UID 等于 UserID 时，才被调度。

B. 将 shell 分离出内核

之前的版本将 shell 写在内核中，现在将其分离，成为一个用户程序。分离时最大的问题是 RunProg 函数的实现，由于 RunProg 函数涉及到很多内核参数，最终的解决方式为通过端口信息传输+信号量的方法，shell 发送信息给内核，内核收到信息后执行程序。

C. 保存和恢复用户的屏幕信息与光标位置

这个比较简单，将段地址 0xB800 的 80*25 个双字节拷贝到当前用户对应的储存屏幕数据的内存段，保存光标位置，然后切换用户（更改 UserID），将新用户原屏幕数据恢复到段 0xB800 处，恢复光标位置。储存屏幕数据的内存段，在 kernel.asm 中分配，使用与内核程序不一样的段（在 C++ 中用 char 数组分配会导致链接出错，原因是链接时段越界）

D. 检测切换终端的按键

检测在内核中实现，当检测到对应按键时，更改 UserID，并做屏幕信息、光标位置的保存与恢复操作。

(2) FAT12 文件系统的文件读写

A. 文件读取

读取的过程为：

1. 从根目录找到对应的文件项，得到文件大小以及文件存储的第一个扇区。
- 2 读取数据区中对应扇区的数据
- 3 查找 FAT，如果值大于等于 0xFF8，说明读取文件已经结束；否则为文件下一个扇区的扇区号，重复过程 2，直至读完。需要注意，文件所在的最后一个扇区，数据可能不占满一个扇区。

B. 文件写入

1. 在根目录查找文件，若文件存在，打开文件；若文件不存在，创建新的文件（在根目录加上新的一项）
- 2 在当前处理的扇区写入数据，如果扇区被填满，申请新的数据区扇区。申请方式为，从头开始查找空闲的 FAT 项，项为 0x000 的代表该项为空，对应扇区即为空闲扇区。更新 FAT，让原扇区对应的 FAT 项等于新扇区的 FAT 编号，新扇区的 FAT 项为 0xFFF，代表文件结束。重复过程 2，直至文件完全写入。
- 3 注意事项：更新 FAT 时，要使用位运算，根据编号的奇偶性更新 1.5 字节的 FAT 项。

(3) FAT12 文件系统的文件操作

A. rm 操作

找到对应项，将文件名的第一个字符改为 0xE5，即标记被删除（可恢复）

B. mv 操作

找到对应项，修改文件名

C. cp 操作

找到对应项，获取文件存储的第一个扇区，生成新的 Entry 并逐扇区写入新的空闲扇区，同时更新 FAT，最终更新写入时间并将 Entry 写入根目录。

五、实验操作

多终端：

按下 Alt + 1, Alt + 2, Alt + 3, Alt + 4 可以切换用户（终端）

在终端 1 下输入 ls 等命令：

Filename	Size	Date	Time(UTC+8)	Clusters
KERNEL.BIN	20680	May.19,2016	21:03:06	3,4,5,6,7,8,...
SHELL.COM	28194	May.19,2016	21:03:08	44,45,46,47,48,49,...
WKCN1.COM	512	May.19,2016	21:03:08	100
WKCN2.COM	512	May.19,2016	21:03:08	101
WKCN3.COM	512	May.19,2016	21:03:08	102
WKCN4.COM	512	May.19,2016	21:03:08	103
KAN.COM	512	May.19,2016	21:03:08	104
BS.COM	16454	May.19,2016	21:03:08	105,106,107,108,109,110,...
BPB.COM	4991	May.19,2016	21:03:08	138,139,140,141,142,143,...
LS.COM	18833	May.19,2016	21:03:08	148,149,150,151,152,153,...
HELP.COM	3939	May.19,2016	21:03:08	185,186,187,188,189,190,...
BOX.COM	327	May.19,2016	21:03:08	193
TESTPORT.COM	2371	May.19,2016	21:03:08	194,195,196,197,198
TESTNEW.COM	9984	May.19,2016	21:03:08	199,200,201,202,203,204,...
FORK1.COM	10256	May.19,2016	21:03:08	219,220,221,222,223,224,...
FORK2.COM	10203	May.19,2016	21:03:08	240,241,242,243,244,245,...
ALPHA.COM	10332	May.19,2016	21:03:08	260,261,262,263,264,265,...
MAT.COM	11501	May.19,2016	21:03:08	281,282,283,284,285,286,...
LETTER.COM	10748	May.19,2016	21:03:08	304,305,306,307,308,309,...
FRUIT.COM	11439	May.19,2016	21:03:08	325,326,327,328,329,330,...
SCREEN.COM	17696	May.19,2016	21:03:08	348,349,350,351,352,353,... Ouch!

```

wkc > uid
UserID: 1
wkc >

```

Ouch! Ouch!

可以看到，UserID = 1

按 ALT + 2，切换到终端 2，执行用户程序 1 和 2

```

WuKan' Program 2
C
D
E
F
G
H
I
J
K
L *
A

```

Ouch! Ouch!

再次输入 ALT + 1，切换为用户 1

Filename	Size	Date	Time(UTC+8)	Clusters
KERNEL.BIN	20680	May.19,2016	21:03:06	3,4,5,6,7,8,...
SHELL.COM	28194	May.19,2016	21:03:08	44,45,46,47,48,49,...
WKC1.COM	512	May.19,2016	21:03:08	100
WKC2.COM	512	May.19,2016	21:03:08	101
WKC3.COM	512	May.19,2016	21:03:08	102
WKC4.COM	512	May.19,2016	21:03:08	103
KAN.COM	512	May.19,2016	21:03:08	104
BS.COM	16454	May.19,2016	21:03:08	105,106,107,108,109,110,...
BPB.COM	4991	May.19,2016	21:03:08	138,139,140,141,142,143,...
LS.COM	18833	May.19,2016	21:03:08	148,149,150,151,152,153,...
HELP.COM	3939	May.19,2016	21:03:08	185,186,187,188,189,190,...
BOX.COM	327	May.19,2016	21:03:08	193
TESTPORT.COM	2371	May.19,2016	21:03:08	194,195,196,197,198
TESTNEW.COM	9984	May.19,2016	21:03:08	199,200,201,202,203,204,...
FORK1.COM	10256	May.19,2016	21:03:08	219,220,221,222,223,224,...
FORK2.COM	10203	May.19,2016	21:03:08	240,241,242,243,244,245,...
ALPHA.COM	10332	May.19,2016	21:03:08	260,261,262,263,264,265,...
MAT.COM	11501	May.19,2016	21:03:08	281,282,283,284,285,286,...
LETTER.COM	10748	May.19,2016	21:03:08	304,305,306,307,308,309,...
FRUIT.COM	11439	May.19,2016	21:03:08	325,326,327,328,329,330,...
SCREEN.COM	17696	May.19,2016	21:03:08	348,349,350,351,352,353,...

Wkc > uid
UserID: 1
Wkc >

按 ALT + 2, 切换到用户 2

```

WuKan's Pro ram 1
C *
D
E
F
G
H
I
J
K
L
M
P
Q
R
S
T
A

```

可以看到，不同终端运行的程序互不影响:-)

使用 topa 可以查看所有用户执行的进程：切换到终端 3 (Alt + 3), 输入 topa

```
MiraiOS 0.7 #1195
You can input 'help' to get more info
wkcncn >
wkcncn > topa
There are 7 Progresses :-)
```

PID	Name	PR	Size	SEG	CS	IP	Parent	State
0	Mirai-Shell	0	0	0x0000	0x0000	0x875A	0	Running
1	SHELL.COM	0	28194	0x46B0	0x0000	0xB692	0	Running
2	SHELL.COM	0	28194	0x4DA3	0x0000	0xB692	0	Running
3	SHELL.COM	0	28194	0x5496	0xF000	0xB70B	0	Running
4	SHELL.COM	0	28194	0x5B89	0x5B89	0x0100	0	Ready
5	WKCNCN1.COM	0	512	0x627C	0x627C	0x0198	0	Running
6	WKCNCN2.COM	0	512	0x62AC	0x62AC	0x017A	0	Running

```
wkcncn > uid
UserID: 3
wkcncn >
```

可以看到 4 个 shell 进程，代表 4 个终端，PID 为 5 和 6 的进程为用户 2 的进程（这里没有显示 UID），4 号 shell 状态为 Ready，说明其还没有被使用过。这里的 state 指的是当 UserID 等于它们的 UID 时的 state.

文件读写：

输入 screen，将当前屏幕信息保存到 screen.txt 中

```
MiraiOS 0.7 #1195
You can input 'help' to get more info
wkcncn >
wkcncn > topa
There are 7 Progresses :-)
```

PID	Name	PR	Size	SEG	CS	IP	Parent	State
0	Mirai-Shell	0	0	0x0000	0x0000	0x875A	0	Running
1	SHELL.COM	0	28194	0x46B0	0x0000	0xB692	0	Running
2	SHELL.COM	0	28194	0x4DA3	0x0000	0xB692	0	Running
3	SHELL.COM	0	28194	0x5496	0xF000	0xB70B	0	Running
4	SHELL.COM	0	28194	0x5B89	0x5B89	0x0100	0	Ready
5	WKCNCN1.COM	0	512	0x627C	0x627C	0x0198	0	Running
6	WKCNCN2.COM	0	512	0x62AC	0x62AC	0x017A	0	Running

```
wkcncn > uid
UserID: 3
wkcncn > screen
wkcncn >
```

切换到终端 4,

open scree: 打开 screen.txt

file 查看文件状态

read 读取文件，每次最多读取 1024 个字节，有 maxBufferSize 决定

```
MiraiOS 0.7 #1195
You can input 'help' to get more info
wkcncn > open screen
Open File: screen.txt successfully!
wkcncn > file
Filename: screen.txt      size: 2000   g: 0   p: 0
wkcncn > read
MiraiOS 0.7 #1195
You can input 'help' to get more info
wkcncn >
wkcncn > topa
There are 7 Progresses :-)
```

PID	Name	PR	Size	SEG	CS	IP	Parent	State
0	Mirai-Shell	0	0	0x0000	0x0000	0x875A	0	Running
1	SHELL.COM	0	28194	0x46B0	0x0000	0xB692	0	Running
2	SHELL.COM	0	28194	0x4DA3	0x0000	0xB692	0	Running
3	SHELL.COM	0	28194	0x5496	0xF000	0xB70B	0	Running
4	SHELL.COM	0	28194	0x5B89	0x5B89	0x0100	0	Ready
5	WKCNC1.COM	0	512	0x627C	0x627C	0x0198	0	Running
6	WKCNC2.COM	0	512	0x62AC	0x62AC	0x017A	0	Running

```
wkcncn > _

wkcncn > topa
There are 7 Progresses :-)
```

PID	Name	PR	Size	SEG	CS	IP	Parent	State
0	Mirai-Shell	0	0	0x0000	0x0000	0x875A	0	Running
1	SHELL.COM	0	28194	0x46B0	0x0000	0xB692	0	Running
2	SHELL.COM	0	28194	0x4DA3	0x0000	0xB692	0	Running
3	SHELL.COM	0	28194	0x5496	0xF000	0xB70B	0	Running
4	SHELL.COM	0	28194	0x5B89	0x5B89	0x0100	0	Ready
5	WKCNC1.COM	0	512	0x627C	0x627C	0x0198	0	Running
6	WKCNC2.COM	0	512	0x62AC	0x62AC	0x017A	0	Running

```
wkcncn > read
Running
wkcncn > uid
UserID: 3
wkcncn > screen

Ouch! Ouch!
wkcncn >
```

可以看到，屏幕信息被正确保存并且能够读写 screen.

测试 seekg, seekp

输入错误序列 abcfedghijk

再进行更正

```

MiraiOS 0.7 #1213
You can input 'help' to get more info
wkcN > open wk
wk.txt not found, create it when writting :-)
wkcN > write abcfedghijk
Written :-)
wkcN > read
abcfedghijk
wkcN > seekp 3
wkcN > write def
Written :-)
wkcN > seekg 0
wkcN > read
abcdefghijk
wkcN > _

```

挂载镜像后，也可以看见创建了一个叫 wk.txt 的文本文件。

测试文件操作

创建一个新文件 test.txt，将其拷贝得到 t.txt

```

wkcN > open test
test.txt not found, create it when writting :-)
wkcN > write test
Written :-)
wkcN > cp test t
wkcN >

```

ls

WK.TXT	11	May.20,2016 11:57:48	399
TEST.TXT	4	May.20,2016 12:10:18	401
T.TXT	4	May.20,2016 12:10:24	402

重命名 t.txt 为 k.txt

```

wkcN > mv t k

```

ls

WK.TXT	11
TEST.TXT	4
K.TXT	4

删除 test.txt 和 k.txt

```

wkcN > rm test
wkcN > rm k

```

ls

```
FRUIT.COM      11439 May.20,2016 11:56:54    338,339
SCREEN.COM     19328 May.20,2016 11:56:54    361,362
WK.TXT         11      May.20,2016 11:57:48     399
wkc >
```

文件被删除

六、小结

本次实验，我实现了多终端和文件读写及操作。

多终端的实现，困难之处在于将 shell 移除内核后，如何让 shell 根据指令执行程序，这涉及到进程间通信。我之前也采取过其它方式，比如把执行程序的函数写成系统中断，问题在于传递字符串比较困难，实现起来很复杂。另一种方式是将执行程序的函数放进一个头文件，问题在于读磁盘时栈不够大，并且涉及很多内核参数。

多终端的另一个问题是保存屏幕数据的内存的位置，最好的方式是另外开辟一个段，保证内核代码的偏移量不超过 64k。

对于文件读写，我的方法是先用 C++ 在 Linux 系统下实现 FAT12 文件系统的读写，然后移植到我的操作系统。在 Linux 系统下，我的主要问题为位运算写错了，导致写入 FAT 时出错。在我的操作系统下，主要问题为 16 位 NASM 与 32 位 G++ 结合，C++ 的结构体无法使用内部函数（由于涉及到压栈），栈空间过小，不能在用户程序内通过栈获得缓冲区。我开始采用了进程通信的方法（include/os_msg.h），无奈出现了错误。由于时间原因，我最终采取的方法是将缓冲区写在数据段内，而不是使用栈。

这次实验，我也了解了宏内核与微内核的区别以及各自的实现方式。我觉得我在操作系统的实验中已经很尽力了，从开始记录编译次数到现在，总共编译了 1213 次，不包括 reset 代码时的次数。我学到了很多，还有更多的东西需要探索～

附上 Github 的提交记录：

刚好一百次提交



