Lab2 系统调用

181860044 李翰

邮箱： 181860044@smail.nju.edu.cn

1. 实验进度：完成了所有内容

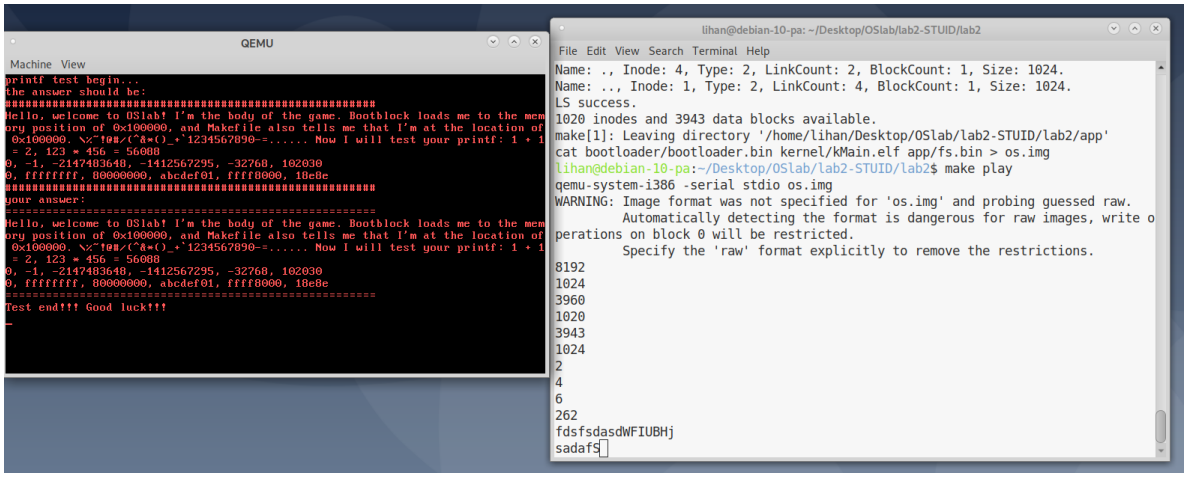
2. 实验结果：

2.1 格式化程序

以下为格式化文件系统时的一些输出信息：

```
lihan@debian-10-pa: ~/D
File  Edit  View  Search  Terminal  Help
gcc -m32 -march=i386 -static -fno-builtin -fno-stack-protector -fno-omit-frame-pointer -Wall
ld -m elf_i386 -e uEntry -Ttext 0x00000000 -o uMain.elf ./main.o ../lib/syscall.o
format fs.bin -s 8192 -b 2
FORMAT success.
1023 inodes and 3959 data blocks available.
mkdir /boot
MKDIR success.
1022 inodes and 3958 data blocks available.
touch /boot/initrd
TOUCH success.
1021 inodes and 3958 data blocks available.
cp uMain.elf /boot/initrd
CP success.
1021 inodes and 3944 data blocks available.
mkdir /usr
MKDIR success.
1020 inodes and 3943 data blocks available.
ls /
Name: ., Inode: 1, Type: 2, LinkCount: 4, BlockCount: 1, Size: 1024.
Name: .., Inode: 1, Type: 2, LinkCount: 4, BlockCount: 1, Size: 1024.
Name: boot, Inode: 2, Type: 2, LinkCount: 2, BlockCount: 1, Size: 1024.
Name: usr, Inode: 4, Type: 2, LinkCount: 2, BlockCount: 1, Size: 1024.
LS success.
1020 inodes and 3943 data blocks available.
ls /boot
Name: ., Inode: 2, Type: 2, LinkCount: 2, BlockCount: 1, Size: 1024.
Name: .., Inode: 1, Type: 2, LinkCount: 4, BlockCount: 1, Size: 1024.
Name: initrd, Inode: 3, Type: 1, LinkCount: 1, BlockCount: 14, Size: 13544.
LS success.
1020 inodes and 3943 data blocks available.
ls /usr
Name: ., Inode: 4, Type: 2, LinkCount: 2, BlockCount: 1, Size: 1024.
Name: .., Inode: 1, Type: 2, LinkCount: 4, BlockCount: 1, Size: 1024.
LS success.
1020 inodes and 3943 data blocks available.
make[1]: Leaving directory '/home/lihan/Desktop/OSlab/lab2-STUID/lab2/app'
cat bootloader/bootloader.bin kernel/kMain.elf app/fs.bin > os.img
lihan@debian-10-pa:~/Desktop/OSlab/lab2-STUID/lab2$ make play
```

2.2 键盘按键串口回显&系统调用库函数printf



3. 代码修改（框架代码中TODO in lab2部分）

3.1 格式化程序

格式化磁盘的相关代码均在 `lab2/utils/genFS` 目录下，这部分仅需完成 `genFS/func.c` 中的 `cp` 函数以将宿主机上的文件拷贝到 `boot` 目录下。

3.2 键盘按键的串口回显

为了加上键盘中断的处理，首先需要在 `lab2/kernel/kernel/idt.c` 中的 `initIdt` 函数中加上对应的门描述符；然后完成 `lab2/kernel/kernel/irqHandle.c` 中的 `keyboardHandle` 函数，完成键盘按键的串口回显。

3.3 实现 `printf` 的处理例程

该部分仅需填充完成 `lab2/kernel/kernel/irqHandle.c` 中的 `syscallPrint` 函数。

3.4 完善 `printf` 的格式化输出

框架代码 `lab2/lib/syscall.c` 中已经提供了 `printf` 最基本的功能，补充其中 `printf` 函数即可。

4.思考&心得体会

1. 内存分页机制将内存地址空间划分为若干个较小的大小固定的页，其实与ext文件系统中将磁盘划分成若干个块相似，分别是为了提高内存和磁盘的利用率。至于ext文件系统并没有采用类似于分页机制中通过固定的两级索引来寻找物理页的方式，个人觉得可能是因为在ext文件系统中，块并不需要顺序存放，且可能需要更多级索引（如三级、四级索引），若类似于分页机制，可能会不便于管理且效率不高。
2. block 与文件大小的关系：
   ext文件系统中 inode 记录 block 号码的区域被定义为12个0级索引，以及各一个1、2、3级索引，所以当一个 block 大小为 1KB 时，相应的一个文件最大大小即可由一个 inode 指向的所有

block 总数求出：（由于记录一个block的索引需要 *4 bytes*，所以 *1KB* 的 block 可以记录 256 个索引）

$$12 \times 1K + 256 \times 1K + 256 \times 256 \times 1K + 256 \times 256 \times 256 \times 1K = 16GB$$

类似的可求得 block 大小为 *2KB* 和 *4KB* 时对应的一个文件最大大小为 *256GB* 和 *2TB*.

3. TSS中没有ring3的堆栈：这是因为只有从较低特权级切换到较高特权级时才需要从TSS中取出相应的堆栈位置信息进行切换，而ring3处于最低特权级，所以TSS中没有ring3的堆栈。

4. 如果在使用 `eax`, `ecx` 等寄存器之前没有将其值保存在栈中，则从内核返回后会产生不可恢复的错误，因为进入中断处理程序时没有保存现场，此时已经无法继续执行产生中断时的程序。

5. 在完善 `printf` 的格式化输出时，发现起初自己的输出并不与正确答案一致，后来查阅MSDN相关文档得知，在 `printf` 函数中 `%` 用于格式控制，与普通字符并不一样，当连续两个 format 字符为 `%` 时，第一个 `%` 实际上相当于转义字符，所以此时只需要输出一个 `%`.

6. 在实现 `printf` 函数时，感觉框架代码中定义的变量 `state` 似乎没有存在的必要，后来想想觉得应该只是实现方式的原因；同时也想到一个问题——如果用于表示格式控制的 `%` 的数量与可变参数数量不同，如何判断可变参数数量是多了还是少了？在网上查到很多相关资料都是说在固定参数中进行设定，即根据 `%` 确定可变参数数量；自己觉得可能是预先设定了一个特殊的结束标志，在将所有可变参数压入栈后将其压入，取参数时访问到该标志便知道参数已经取完。后来自己编写代码利用 `objdump` 工具进行测试，发现 `printf` 函数可能确实是根据固定参数 `format` 来分析可变参数，并没有实现智能识别，且编译器对可变参数的数量是否与 `%` 匹配也不做严格检查。于是也意识到这是 `printf` 函数的一个不安全之处，从而也对编程人员提出了较高的要求。

7. 本次实验使得本人了解了 ext 文件系统的一些基础知识、一个格式化程序大概需要做什么以及如何用代码实现之，进一步理解了基于中断实现系统调用全过程；同时通过对 `printf` 函数的实现和思考，也加深了对该函数的认识。虽然实验中涉及的需要补充的代码并不算很多，但阅读并理解框架代码的过程也十分重要，这也是对课本理论知识的实现以及运用。