

Lab2

181860154 朱倩

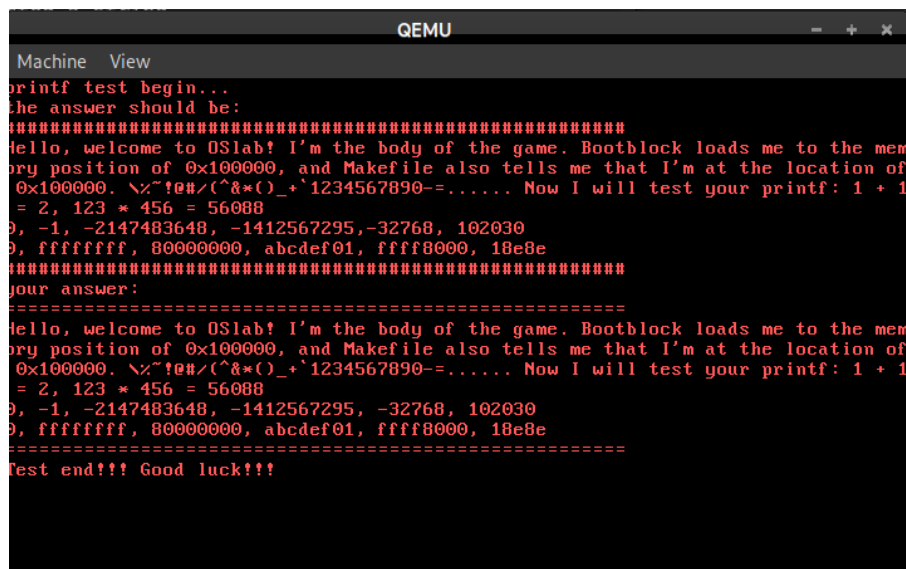
邮箱: infinite0124@163.com

一、实验进度

完成了所有内容

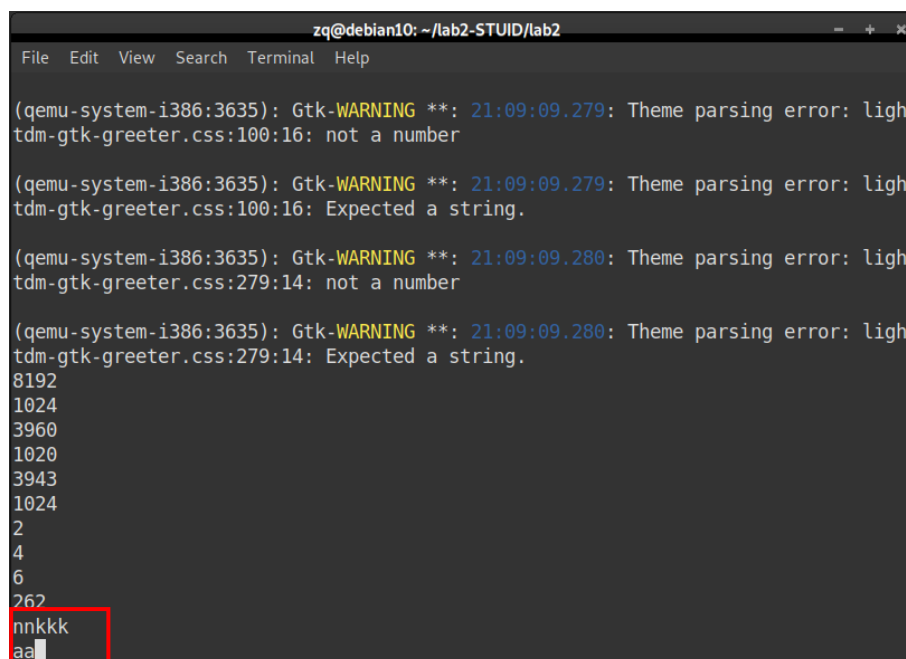
二、实验结果

Printf的输出结果:



```
Machine View
printf test begin...
the answer should be:
#####
Hello, welcome to OSlab! I'm the body of the game. Bootblock loads me to the mem
ory position of 0x100000, and Makefile also tells me that I'm at the location of
0x100000. \x"!@#/(^&*()_+'1234567890-=..... Now I will test your printf: 1 + 1
= 2, 123 * 456 = 56088
0, -1, -2147483648, -1412567295, -32768, 102030
0, ffffffff, 80000000, abcdef01, ffff8000, 18e8e
#####
your answer:
=====
Hello, welcome to OSlab! I'm the body of the game. Bootblock loads me to the mem
ory position of 0x100000, and Makefile also tells me that I'm at the location of
0x100000. \x"!@#/(^&*()_+'1234567890-=..... Now I will test your printf: 1 + 1
= 2, 123 * 456 = 56088
0, -1, -2147483648, -1412567295, -32768, 102030
0, ffffffff, 80000000, abcdef01, ffff8000, 18e8e
=====
Test end!!! Good luck!!!
```

键盘输入的结果:



```
zq@debian10: ~/lab2-STUID/lab2
File Edit View Search Terminal Help

(qemu-system-i386:3635): Gtk-WARNING **: 21:09:09.279: Theme parsing error: ligh
tdm-gtk-greeter.css:100:16: not a number

(qemu-system-i386:3635): Gtk-WARNING **: 21:09:09.279: Theme parsing error: ligh
tdm-gtk-greeter.css:100:16: Expected a string.

(qemu-system-i386:3635): Gtk-WARNING **: 21:09:09.280: Theme parsing error: ligh
tdm-gtk-greeter.css:279:14: not a number

(qemu-system-i386:3635): Gtk-WARNING **: 21:09:09.280: Theme parsing error: ligh
tdm-gtk-greeter.css:279:14: Expected a string.
8192
1024
3960
1020
3943
1024
2
4
6
262
nnkkk
aa
```

三、 实验修改的代码位置

utils/genFS/func.c:

int cp(...): 完成该函数新建一个文件并把指定文件内容拷贝到该文件的功能

kernel/kernel/idt.c:

void initIdt(): 增加键盘中断对应门描述符的设置

kernel/kernel/irqHandle.c:

void keyboardHandle(...): 利用键盘驱动接口和串口输出接口完成键盘按键串口回显

void syscallPrint(...): 将指定字符串写显存

lib/syscall.c:

void printf(...): 完成printf中解析参数的过程

app/main.c:

int uEntry(void): 增加printf的测试代码

四、 思考题

1. 分别考虑内存分段机制, ext 文件系统和内存分页机制, 它们之间是否有某种联系? 回忆 i386 分页机制, 为什么 ext 文件系统没有采用类似于分页机制中对物理页的组织方式, 通过固定的两级索引来寻找物理页?

答: 都采用了将存储空间划分为一定单位大小, 将数据分散存储在相应数量的单位, 记录实际数据存储位置的编号, 通过编号索引来访问数据的机制。

因为物理页一般是很大的, 所以用二级目录来避免目录占用较大连续空间的情况。而文件大多数相比于物理页是比较小的, 目录并不会占用很多空间, 需要用到两级索引的大文件是少数, 所以文件系统并没有固定使用两级索引。

2. 对目录的硬链接: Linux 不允许对目录创建硬链接, 你知道为什么吗? 如果允许对目录创建硬链接, 你能否举出一个影响系统工作的例子?

答: 允许对目录创建硬链接很容易引起文件系统的混乱。比如将父目录下的一个子目录链接到父目录, 将会造成目录的循环, 很容易出错。

3. 保存寄存器的旧值: 我们在使用eax, ecx, edx, ebx, esi, edi 前将寄存器的值保存到了栈中, 如果去掉保存和恢复的步骤, 从内核返回之后会不会产生不可恢复的错误?

答: 会的, 程序运行时常常会把一些关键值存放在寄存器中, 进入内核态之前如果不保存现场, 或是从内核返回之后不恢复现场, 都会导致原程序运行时的某些关键数值丢失或出错, 造成程序的运行错误。

—Thanks for reading! —