Oslab2系统调用实验报告

**一．实验目的**

1.学习基于中断的完整的系统调用机制

2.了解并实现printf、scanf实现原理

3.熟悉底层的io实现过程

1. **实验设计**

1.程序的流程是：

（1）Bootloader从实模式进入保护模式，加载内核至内存，跳转执行

（2）内核初始化IDT（中断描述符表），初始化GDT（全局描述符表），初始化TSS（任务状态段）

（3）内核加载用户程序至内存，对内核堆栈进行设置，通过iret切换至用户空间，执行用户程序

（4）用户程序调用库函数scanf和printf陷入内核，由内核完成读取键盘输入，在视频映射的显存地址中写入完成字符串的打印等内容

（5）完成测试用例

2.需要完成的任务：

（1）lib/syscall.c的printf和scanf

（2）kernel/kernel/irqHandle.c中的syscallScan

3.代码思路

（1）printf

函数框架将printf第一个参数format逐个读入；

如果不是特定的token%则输入到buffer中，则每当buffer满了则调用syscall写接口到std\_out；

实现token的思路利用switch语句检查%后的关键词，然后调用对应接口（框架代码中已给出）处理，

其中需要思考的是如何获取函数的更多参数，利用栈的原理，将指针paraList从format依次增4个字节即可访问对应的参数，因为函数的参数是从高到低压栈，参数列表按从右到左顺序。利用指针类型转换可以实现取到确定地址所需类型的变量！index表示解析的token个数



（2）scanf

和printf函数实现功能类似，首先逐步读入第一个参数对应数组format，

每当buffer是空，则调用syscall的读接口到std\_in,如果format中不是%的token，则要在buffer中严格匹配普通字符，否则返回-1；如果是format有空格则对应跳过buffer中的空格，如果解析到%，则利用switch语句查看解析类型，然后调用特定接口（框架中已给出的函数）。循环在需要的地方使用buffer中的数据给传入的参数，参数获取也是利用指针在栈帧中不断加4字节获取，和printf不同的是scanf里参数全是地址，形式上要利用例如\*（void\*\*）的形式取得对应位置的指针值。

在switch外，单独实现了%？s的功能，？表示一个正整数，首先计算format中？的取值（循环即可）

然后调用对应的接口，这个正整数k表示读入字符串的长度，k+1表示包括结束符‘\0‘的长度，作为参数avail传入string2String2

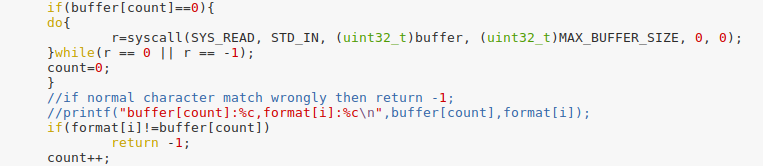
Swtich实现截图



%ks实现截图



严格匹配实现截图：



（3）对syscall.c框架的API函数大致理解

dec2Str:传入一个int，打印

hex2Str：传入一个uint32\_t，按16进制打印

str2Str:传入一个字符串，打印

matchWhiteSpace:跳过buffer中目前所指向的连续的空格，直到下一个有效值或结束符为止

str2Dec:传入buffer作为接受缓冲数组，写入一个int\*指针所指向的区域

str2Hex:传入buffer作为接受缓冲数组，写入一个int\*指针所指向的区域（在bit级别上int和uint\_32并无差别）

str2Str2:传入一个buffer作为接受缓冲数组，传入avail为包括结束符在内的有效写入长度，写入一个char\*（即string）指针所指向的区域

（4）syscallScan

按照提示，需要利用getKeyCode（实现在keyboard.c中）获取当前的key code

然后把keycode循环写入keyBuffer中

这里有几个要注意的地方（坑）

第一．

getKeyCode函数的实现当没有键盘输入的时候默认返回0，这是在阅读keyboard.c发现的

所以在读入之前要先判断键码是否不为0，才能写入keybuffer

第二

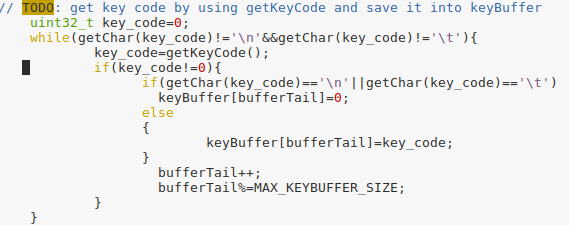
这里键码不需要知道具体的值，只需要getChar（键码）后判断对应字符取值即可

第三

Buffertail的加法是模加法，这个容易忽略

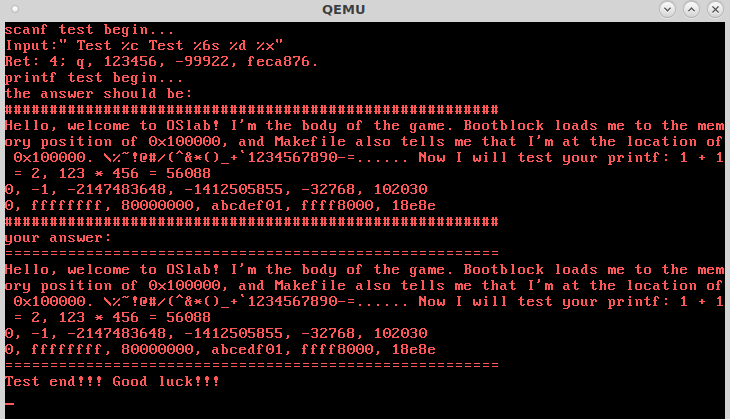
在注意到上述三点后只要循环读keybuffer即可，直到接收到一个换行符或制表符，然后返回即可

下为代码实现截图



1. 实验结果

Make clean、make、make play运行代码获取如下结果（输入仅供参考）：



对应输入流：

C:\Users\吴紫航\AppData\Roaming\Tencent\Users\401986905\QQ\WinTemp\RichOle\OE9ZL479QEG3Q~$DI6K4211.png

1. 实验心得
2. 在实验感到无从下手的时候，将框架代码每个文件的关系和功能整理清楚会很有帮助

比如我在实现syscallScan的时候，刚开始很茫然，后来仔细阅读框架代码，发现了keyboard中有getChar、getKeyCode函数的代码，阅读后马上有了思路。

1. 按照框架给定函数的思路组织自己的代码会非常方便

本次实验给的接口极大减少了自己的代码量，当然框架API也有一些错误，比如api的string2Dec、string2Hexadec，读入syscall在最开始循环前调用一次就可以了，不必在循环中反复检查buffer是否空然后调用，因为读取整型在无效字符（分隔符）出现就结束，循环体中调用syscall会导致最后一次循环（将要完成读入的时候）检测到buffer是空，又要进行输入，与预期要求不符。

1. 实验建议

暂无