lab2 实验报告

李培佳 191300029 191300029@smail.nju.edu.cn

1.实验进度

完成了所有实验内容，即中断机制，系统调用库函数`printf`和对应的处理例程，键盘按键的串口回显，系统调用库函数`getChar`、`getStr`和对应的处理例程。

2.实验结果

3.修改的代码位置

3.1 中断机制

为了进入内核，首先需要在`/bootloader/start.s`中设置`esp`，查看`/kernel/kernel/kvm.c`可以看到内核的`esp`初始值应该是`0x1fffff`，然后在`/bootloader/boot.c`中的`bootmain`函数中设置函数入口，进入内核空间。

内核空间中的一系列初始化，分别对应一系列函数。

> 1. 初始化串口输出
> 2. 初始化中断向量表（`initIdt`）
> 3. 初始化`8259a`中断控制器（`initIntr`）
> 4. 初始化`GDT`表、配置`TSS`段（`initSeg`）
> 5. 初始化VGA设备（`initVga`）
> 6. 配置好键盘映射表（`initKeyTable`）
> 7. 从磁盘加载用户程序到内存相应地址（`loadUMain`）
> 8. 进入用户空间（`enterUserSpace`）

- `initIdt`中有两个函数分别用于设置中断门和陷阱门，根据传入的参数对结构体的内容进行修改即可，在`initIdt`函数中，根据框架代码中标出的中断号，使用上述的中断门和陷阱门函数进行设置。
- 其余初始化函数在框架代码中已经写好。
- `enterUserSpace`在`loadUMain`中，和进入内核空间的方法一致，不同的是elf文件的地址不同，用户空间为`0x200000`

3.2 `Print`系统调用

当执行到`printf`函数时，首先处理参数，当传入的参数`format`还没有处理到最后一个字符时，始终进行循环。

对于普通字符，直接将其传入`buffer`中即可，而一旦遇到`%`，说明需要格式化输出，我们继续向前移动一位来判断格式是什么，本次实验要求实现`%d %x %s %c`四种输出，故`printf`大致框架如下

> switch(format):
>
> case 'd' : 先处理成10进制数字，再转化为字符串
>
> case 'x' : 先处理成16进制数字，再转化为字符串

> case 'c' : 输出字符
>
> case 's' : 输出字符串

当`printf`循环结束后，我们得到了一个`buffer`，里面存放着最终我们要打印到屏幕上的内容。这时进行系统调用`SYS_OUT, STD_OUT`，产生中断，进入内核态，经过`irqhandle`的处理最终进入`syscallPrint`函数。

由于访问了用户段的数据，需要设置`sel`为`USEL(SEG_UDATA)`

使用内联汇编将`data`打印到屏幕上的`pos`处

```
asm volatile("movw %0, (%1)"::"r"(data),"r"(pos+0xb8000));
```

关于光标的维护，逻辑清晰即可，即每次打印一个字，列数加一，框架代码中有全局变量`displayRow displayCol`来代表当前光标的位置

判断当前位置，如果是最后一列，则换到下一行第一列，如果是最后一行，则使用`scrollScreen`刷新屏幕使其最下方多出来一行（也即整体向上移一行，第一行被覆盖）

遇到换行符则直接换行。

3.3 键盘按键显示

键盘输入也是一个中断，其中断号为`0x21`，故当键盘输入时，同样会进入内核态，经过`irqhandle`的处理最终进入`KeyboardHandle`函数。根据框架代码的提示，分为退格符，回车符和正常字符。

首先实现正常字符，思路与`syscallPrint`相同，将接受到的字符打印到屏幕上，注意屏蔽不可打印字符，**由于`keyboard.c`中没用定义整个键盘，故有些字符并没有被屏蔽**，如`Delete RCtrl`等

回车符和`syscallPrint`中的`\n`相同，进行换行即可

退格符思路相反，用空字符覆盖前一个即可，由框架代码的要求，在退到第一列时，不再进行退格，即当位置处于第一列时，跳过执行。

3.4 `getChar,getStr`系统调用

`getChar`对应`SYS_READ STD_IN`，中断进入内核后，最终进入`syscallGetChar`函数，一直接收键盘输入，直到接收到非空字符输入，此时继续等待输入，使用一个新的变量去接收输入，如果收到回车符则返回，正常可打印字符不打印（因为已经接收到了`char`），退格符则清空缓存区，重新接收正常字符，最终使用`tf->eax`传递返回值，在`getChar`中接受参数即可。

`getStr`对应`SYS_READ STD_STR`，中断进入内核后，最终进入`syscallGetStr`函数，思路相似，在接收到回车符或者字符串长度达到`size`前，一直接收字符，并把字符串首地址传回`getStr`，由于此思路涉及数据段的切换，一直尝试没有成功，最后选择了在`getStr`中使用`getChar`的方法进行封装，具体为`syscallGetStr`每次只返回一个字符，直到接收到回车符或者长度达到限制。同时`getStr`中也做到了退格，约定接收到退格符时，内核返回一个特殊字符`?`，（

`getChar getStr`均使用了这样的约定）这样用户函数得知需要退格，删除缓冲区字符串中的部分数据。（虽然这样做让用户无法接收`?`）

注意：`keyboard.c`中修改了`getKeyCode`，加入了`old_code`来防止按下一个键后大量的重复输入

关于大小写

第一次完成键盘后可以正常切换大小写，但是当完成所有内容后，发现大小写无法正常切换，具体为`Capslock`不起作用，`RShift`和`LShift`有作用，并且在不同品牌的电脑上运行效果都不相同（经过多台电脑测试得知）

在`getStr getChar`调用过程中，无法做到大小写切换，故实验结果一栏的`bob`只能是小写。

以上是本次实验报告的全部内容，感谢阅读！