南京航空航天大学

操作系统实践

第一次实验

班级： 1618001
学号： 161840230
姓名： 王可

一、 题目简述

1. 添加用户命令(40%)

实现命令pxy Z（xy为学号最后两位），使得屏幕输出OS Lab $学号$: Z 并换行。

2. 添加内核输出语句(10%)

在系统启动输出的信息中，加入自己的学号姓名，具体位置在” cpu: starting 0” 与 “sb : size 1000 nblocks...” 之间。

3. 添加系统调用(50%)

添加用户命令shutdown，由于功能需要调用特权指令outw(0x604,0x2000)，应实现系统调用来完成用户命令，使得输入shutdown后，推出qemu到ssh窗口。

二、 实验过程

1. 添加用户命令

(1) 添加用户命令首先需要定义命令的逻辑，实现一个名为p30.c的文件（文件名与命令名称对应）并通过main函数参数接收Z的内容，首先输出"OS Lab 161840230: "，然后对除命令本身的其他输入依次输出，用空格隔开，最后输出一个换行符。

```
#include "types.h"
#include "stat.h"
#include "user.h"

int main(int argc,char* argv[]){
    printf(1,"OS Lab 161840230: ");
    if(argc > 1){
        for(int i = 1; i < argc;++i){
            printf(1,"%s ",argv[i]);
        }
    }
    printf(1,"\n");
    exit();
}
```

(2) 其次，需要将该用户命令注册到用户命令列表中。 对此，需要对Makefile中的 UPROGS量添加一个_p30\进行注册。

(3) 完成后进行make生产，再次打开系统就可以成功执行命令p30。

2. 添加内核输出语句

(1) 通过阅读源码可以发现，"cpu starting"在mpmain(void)函数的第一行被输出，而通过测试，下一句系统输出在scheduler()函数中，或者之后，因此，在这两者之间任意地方增加语句：

```
cprintf("wangke 161840230\n");
```

即可。

3. 添加系统调用

(1) 首先，添加系统调用语句，命名为sdown，在proc.c中进行编写注册，sdown中调用特权指令outw(0x604,0x2000)。

```
int
sdown(void){
  outw(0x604,0x2000);
  return 22;
}
```

(2) 在user.h,defs.h中注册sdown函数，用于后续绑定(会被其他定义处引用)。

(3) 在syscall.h中将SYS_sdown系统命令与命令号22绑定，作为sdown的返回值，且将此命令号在syscall.c中与系统调用sys_sdown进行绑定，并注册sys_sdown命令。

(3) 在sysproc.c文件中定义 sys_sdown函数，

(4) 按照实验一的方式添加用户命令shutdown，并调用系统调用sdown，即可完成关机指令。

(5) 打开系统，输入shutdown，正常退出。

三、 实验结果

1. 添加用户命令

在git bash中输入make qemu指令，启动xv6，输出如下信息：

```
xv6...
cpu0: starting 0
wangke 161840230
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
init: starting sh
```

2. 添加内核输出语句

输入p30以及一个含空格的字符串，输出指定结果：OS Lab (学号): Z

```
$ p30 hello world
OS Lab 161840230: hello world
```

3. 添加系统调用

在xv6系统中输入shutdown，成功进行关机操作，退出到ubuntu界面：

```
$ shutdown
wangke@VM-199-186-ubuntu:~/proj0-revise$
```

四、 实验心得

1. 添加用户命令时，起初使用return 0返回出错，考虑可能是返回值没有给良好的处理，所以进trap了，改为exit(0)后解决。
2. 考虑内核输出时，和用户空间输出的差别在于，从用户空间输出时，需要中断调用输出，切换状态，切换上下文，需要更多代价，而从内核态输出，只需要函数调用即可，因此速度更快，对计算机资源的浪费更低。 此外，从内核输出，可以减少用户命令的绑定，使得命令空间更加简洁。
3. 操作系统调试，报错时信息量比较少，修改时应该做版本控制或记录修改处。