

# 操作系统实验 LAB2 实验报告

陈泓宇 171830638  
171830638@smail.nju.edu.cn

## 一. 实验内容

### 1. 格式化程序：按照文件系统的数据结构，构建磁盘程序

- (1) 通过观察 Make File 可以看到，在执行 Make 指令时，宿主机首先在 qemu 以外为其格式化一个磁盘；
- (2) 观察框架代码，我们只需要编写一个 copy 函数；
- (3) 与 Mkdir 函数（新建一个文件夹）进行比较，我们可以发现，编写 copy 函数只需要调用 copyData 函数，为其准备参数，并且使用 allocInode 函数修改 superblock、fatherInode、inodebitmap、blockbitmap，以及为新建的文件定义 destInode 即可；
- (4) 调用 copyData 函数如下：

```
ret = copyData(file, fileSrc, &superBlock, &destInode, destInodeOffset);
```

### 2. 键盘按键的串口回显：处理由用户按键造成的中断请求

- (1) 在 IDT 表中增加对应的门描述符，其中键盘中断号为 0x21，调用的处理函数为 irqKeyboard，权限为内核态权限：

```
setIntr(idt + 0x21, SEG_KCODE, (uint32_t)irqKeyboard, DPL_KERN
```

- (2) 编写处理函数 irqKeyboard：

```
void keyboardHandle(struct TrapFrame *tf)
{
    // TODO in lab2
    uint32_t code = getKeyCode();
    char temp = getChar(code);
    putChar(temp);
    return;
}
```

### 3. 实现 printf 的处理例程：vga 模式字符打印

- (1) 用户在用 int 0x80 进行系统调用后，通过不同的系统调用号选择不同的处理例程，此处完善函数 syscallPrint；
- (2) 通过以下代码将字符打印在屏幕上：

```

data = character | (0x0c << 8); //set color and char;
pos = (80*displayRow+displayCol)*2; //set position on vga;
//print char on screen;
asm volatile("movw %0, (%1)":"r"(data),"r"(pos+0xb8000));

```

(3) 对'\n'、换行、滚屏的处理：若检查到字符为'\n'，将 displayCol 置 0，将 displayRow 加 1；当 displayCol>=80 时，将 displayRow 加 1；当 displayRow>=25 时，将 displayRow 直接置为 24（最后一行）并将 displayCol 置 0，调用 scrollScreen 函数；

## 4.完善 printf 的格式化输出：实现格式转换说明符

(1) .printf 格式转换的参数调用原理：在准备 printf 函数的参数时依次对\*format 等进行压栈，所以只需每次将栈中的指针加上该参数的大小，即可得到下一个参数的地址；

(2) 格式转换的说明符为'%c'、'%x'、'%s'、'%d'等，只需在即将加入 buffer 的字符为 '%' 时，用 switch 对下一个字符进行选择，然后分别调用不同的对应的函数，否则直接将该字符加入 buffer；

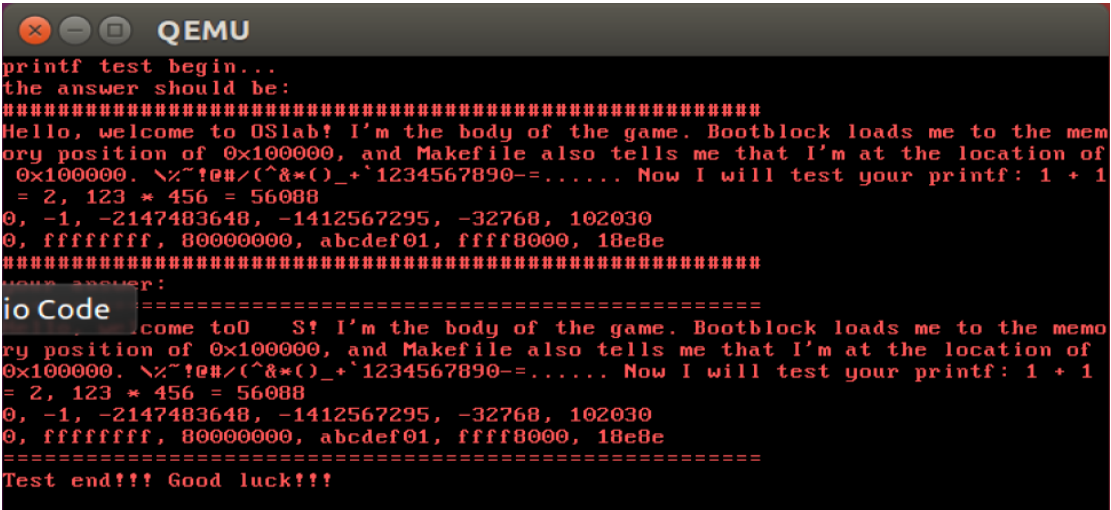
(3) .printf 的核心代码：

```

if (format[i] == '%')
{
    i++;
    switch (format[i])
    {
        case 'd':
            decimal=*(int*)paraList;
            paraList+=sizeof(int);
            count=dec2Str(decimal,buffer,MAX_BUFFER_SIZE,count);
            break;
        case 's':
            string=*(char**)paraList;
            paraList+=sizeof(char*);
            count=str2Str(string,buffer,MAX_BUFFER_SIZE,count);
            break;
        case 'x':
            hexadecimal=*(uint32_t*)paraList;
            paraList+=sizeof(uint32_t);
            count=hex2Str(hexadecimal,buffer,MAX_BUFFER_SIZE,count);
            break;
        case 'c':
            character = *(char *)paraList;
            paraList+=sizeof(char);
            buffer[count++] = character;
            break;
        case '%':
            buffer[count++]='%';
            break;
    }
    i++;
}

```

## 5.实验结果



```
printf test begin...
the answer should be:
#####
Hello, welcome to USlab! 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
#####
io Code:
#####
come to 0x100000, S! I'm the body of the game. Bootblock loads me to the memo
ry 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!!!
```

## 二 . 问题回答

1.分别考虑内存分段机制、ext 文件系统、内存分页机制，他们之间是否有某种联系？回忆 i386 机制，为什么 ext 文件系统没有采取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：内存分段机制、ext 文件系统、内存分页机制都是采用的分级索引的机制；固定的两级索引所能储存的文件大小将变小；

2.计算表中数据？

块大小	1KB	2KB	4KB
一个文件最大大小	16GB	256GB	2TB
文件系统最大大小	4TB	8TB	16TB

答：以 1KB 大小的块为例，三级索引下可以记录的单个文件的区间容量为 256\*256\*256\*1KB=16GB；

3.在磁盘上创建新文件的“幽灵空间”问题？

答：创建的新文件需要建立 inode 结构体，inode 的大小算作文件大小的一部分；

4.linux 系统不允许对目录创建硬链接的原因？若允许创建硬链接，如何影响系统工作？

答：创建硬链接可能会形成目录环；一个目录下的 (.) 和 (..)，即当前目录和父目录即为两个硬链接；

5.若去掉保存和恢复寄存器旧值的过程，会不会导致出错？

答：会，因为调用中断处理程序会改变寄存器的值；