

# OS lab2实验报告

姓名	学号	邮箱	院系
张洋彬	191220169	<a href="mailto:1016466918@qq.com">1016466918@qq.com</a>	计算机科学与技术系

## OS lab2实验报告

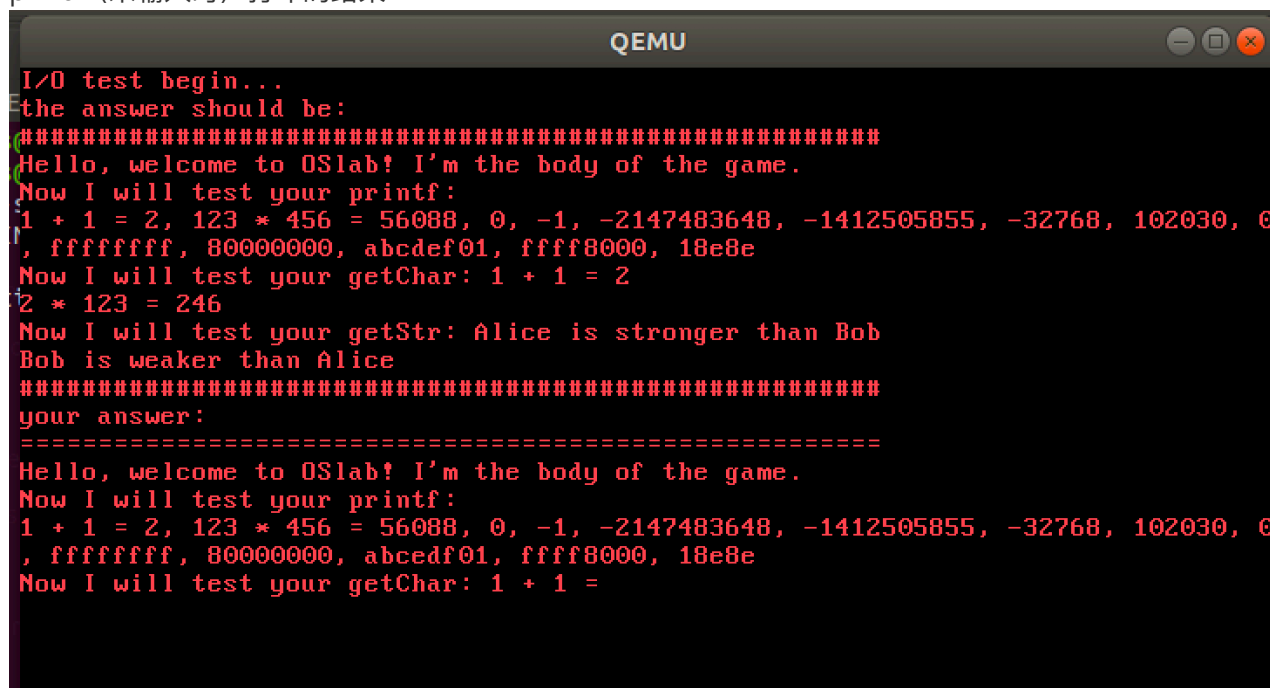
- 一、实验目标
- 二、实验结果
- 三、实验过程
  - 3.1 实现bootmain函数
  - 3.2 初始化
  - 3.3 实现loadUmain函数
  - 3.4 完善中断服务例程
  - 3.5 实现 `syscallPrint` (`printf` 对应的处理例程)
  - 3.6 实现 `printf` 的格式化输出
  - 3.7 实现 `KeyboardHandle()` 函数
  - 3.8 实现 `getchar()` 函数
  - 3.9 实现`getStr()`函数
- 四、实验介绍中的问题
- 五、关于本次实验的感想

## 一、实验目标

- 实现中断机制，完善IDT、TSS、中断处理程序等必要结构。
- 实现系统调用库函数 `printf`
- 实现系统调用库函数 `getChar`
- 实现系统调用库函数 `getStr`

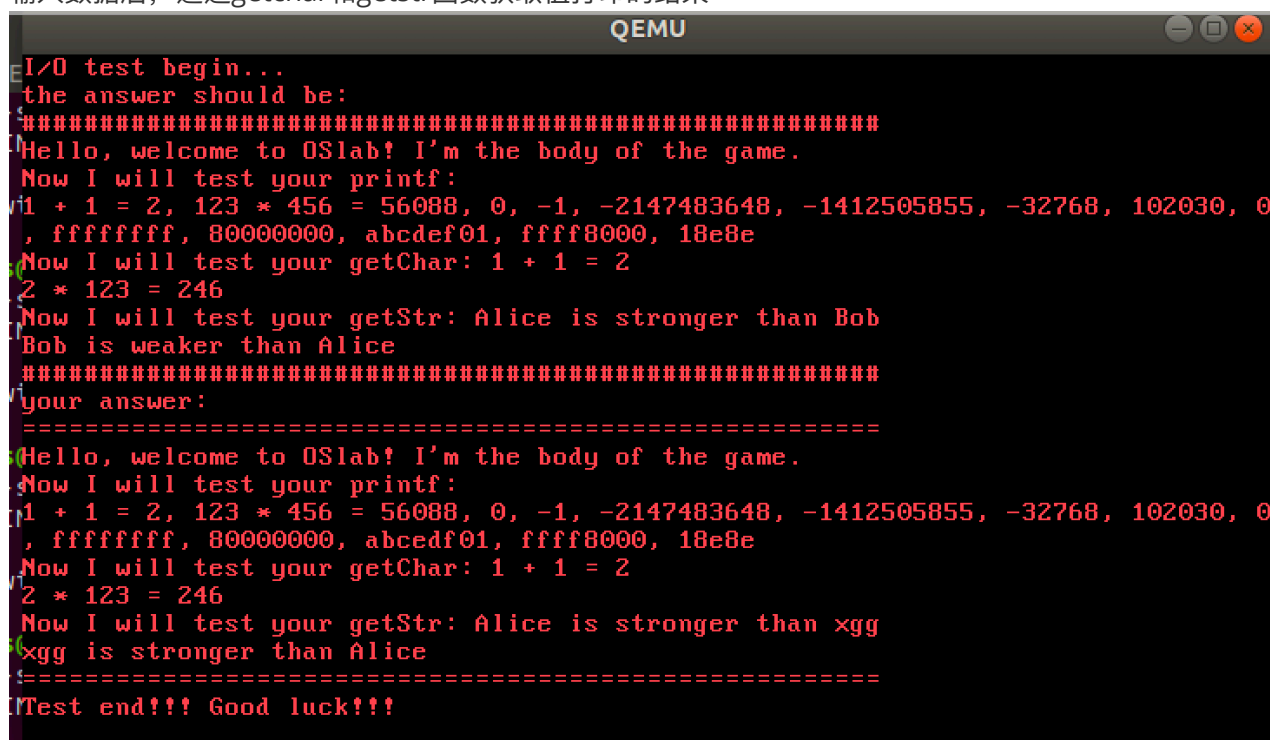
## 二、实验结果

printf (未输入时) 打印的结果



```
QEMU
I/O test begin...
the answer should be:
#####
Hello, welcome to OSlab! I'm the body of the game.
Now I will test your printf:
1 + 1 = 2, 123 * 456 = 56088, 0, -1, -2147483648, -1412505855, -32768, 102030, 0
, ffffffff, 80000000, abcdef01, ffff8000, 18e8e
Now I will test your getChar: 1 + 1 = 2
2 * 123 = 246
Now I will test your getStr: Alice is stronger than Bob
Bob is weaker than Alice
#####
your answer:
=====
Hello, welcome to OSlab! I'm the body of the game.
Now I will test your printf:
1 + 1 = 2, 123 * 456 = 56088, 0, -1, -2147483648, -1412505855, -32768, 102030, 0
, ffffffff, 80000000, abcdef01, ffff8000, 18e8e
Now I will test your getChar: 1 + 1 =
```

输入数据后, 通过getchar和getstr函数获取值打印的结果



```
QEMU
I/O test begin...
the answer should be:
#####
Hello, welcome to OSlab! I'm the body of the game.
Now I will test your printf:
1 + 1 = 2, 123 * 456 = 56088, 0, -1, -2147483648, -1412505855, -32768, 102030, 0
, ffffffff, 80000000, abcdef01, ffff8000, 18e8e
Now I will test your getChar: 1 + 1 = 2
2 * 123 = 246
Now I will test your getStr: Alice is stronger than Bob
Bob is weaker than Alice
#####
your answer:
=====
Hello, welcome to OSlab! I'm the body of the game.
Now I will test your printf:
1 + 1 = 2, 123 * 456 = 56088, 0, -1, -2147483648, -1412505855, -32768, 102030, 0
, ffffffff, 80000000, abcdef01, ffff8000, 18e8e
Now I will test your getChar: 1 + 1 = 2
2 * 123 = 246
Now I will test your getStr: Alice is stronger than xgg
xgg is stronger than Alice
=====
Test end!!! Good luck!!!
```

# 已完成了lab2的所有内容

## 三、实验过程

## 3.1 实现bootmain函数

修改kMainentry、phoff和offset的值。

```
1 unsigned int elf = 0x100000; //内核的elf值
2
3 kMainEntry=(void(*) (void))(((struct ELFHeader*)elf)->entry);
4 phoff=((struct ELFHeader*)elf)->phoff;
5 offset=((struct ProgramHeader*)(elf+phoff))->off; //elf加上phoff等于程序头的偏移量
```

## 3.2 初始化

将函数运行所需要的所有内容进行初始化，初始化内容如下：

```
1 initSerial(); // initialize serial port
2 initIdt();      // initialize idt
3 initIntr();     // initialize 8259a
4 initSeg();      // initialize gdt, tss
5 initVga(); // initialize vga device
6 initKeyTable(); // initialize keyboard device
7 loadUMain(); // load user program, enter user space
```

## 3.3 实现loadUmain函数

在 kernel/kernel/kvm.c 中，仿造bootMain函数来实现loadUmain函数，不同的点是elf头的值以及enterUserspace的方式。

```
1 unsigned int elf = 0x200000; //用户代码的elf值
2 uint32_t uMainEntry = 0x200000;
3 中途跟bootmain类似
4 .....
5 enterUserSpace(uMainEntry);
```

## 3.4 完善中断服务例程

在 kernel/kernel/idt.c 的 initIdr() 函数中添加键断对应的门描述符、中断号、中断处理函数、中断特权级。

```

1  setTrap(idt + 0x8, SEG_KCODE, (uint32_t)irqDoubleFault, DPL_KERN);
2  setTrap(idt + 0xa, SEG_KCODE, (uint32_t)irqInvalidTSS, DPL_KERN);
3  setTrap(idt + 0xb, SEG_KCODE, (uint32_t)irqSegNotPresent, DPL_KERN);
4  setTrap(idt + 0xc, SEG_KCODE, (uint32_t)irqStackSegFault, DPL_KERN);
5  setTrap(idt + 0xd, SEG_KCODE, (uint32_t)irqGProtectFault, DPL_KERN);
6  setTrap(idt + 0xe, SEG_KCODE, (uint32_t)irqPageFault, DPL_KERN);
7  setTrap(idt + 0x11, SEG_KCODE, (uint32_t)irqAlignCheck, DPL_KERN);
8  setTrap(idt + 0x1e, SEG_KCODE, (uint32_t)irqSecException, DPL_KERN);
9  setIntr(idt + 0x21, SEG_KCODE, (uint32_t)irqKeyboard, DPL_KERN);
10 setIntr(idt + 0x80, SEG_KCODE, (uint32_t)irqSyscall, DPL_USER); //用户级别

```

## 3.5 实现 `syscallPrint` (`printf` 对应的处理例程)

在 `kernel/kernel/irqHandle.c` 中完善 `syscallPrint()` 函数, 思路如下:

```

1  asm volatile("movb %%es:(%1), %0"::"r"(character):"r"(str+i)); //把str[i]读到character里
2  if(character=='\n'){
3      displayCol=0;
4      displayRow++;
5      if(displayRow=25) //已经满了, 超出边界
6          行号-1, 调用scrollScreen函数往上滚一行
7  }
8  else{
9      data = character | (0x0c<<8);
10     pos = (80*displayRow + displayCol)*2;
11     asm volatile("movw %0, (%1)"::"r"(data), "r"(pos+0xb8000)); //将character打印出来
12     displayCol=(displayCol+1)%80;
13     if(displayCol==0){
14         ..... //实现如上
15     }
16 }

```

## 3.6 实现 `printf` 的格式化输出

在 `syscall.c` 函数里面已经实现了 `dec2str`, `str2str`, `hex2str` 函数, 在 `printf` 里面需调用他们实现中断, 以 decimal 来进行说明, 代码如下:

```

1  case 'd':
2      index++;
3      decimal = *(int*)(paraList + 4 * index);
4      count = dec2Str(decimal, buffer, (uint32_t) MAX_BUFFER_SIZE,
5      count);
6      break;

```

- 外层是一个 while 循环, 当遇到 `format[i] = '%'` 时表示此处需要格式化输出

- `%d` 表示十进制数，类型为 `int`，现将 `index++` (目前是 `%d`)；通过 `*(int*)(paraList + sizeof(void*) * index)` 取到参数列表中的第 `index` 个参数。并通过强制转换得到 `int` 型数据。
- 利用框架代码提供的 `dec2Str()` 函数将十进制整数转换为字符串并修正 `count`

## 3.7 实现 `keyboardHandle()` 函数

`keyboardhandle` 主要分三种情况，先使用 `getchar(code)` 得到输入字符的 `ascii` 码，然后思路如下：

```

1  if(退格符):
2      displayCol--, 然后用0去覆盖已经打印的字符
3  else if(回车符):
4      和systemPrint实现类似，把回车符存入keyBuffer
5  else if(正常字符):
6      if(code==0x2a || code == 0x36 || code == 0x38 || code == 0x3a || code == 0x1d){
7          keyBuffer[bufferHead]=code;
8      }
9      else{
10         if(keyBuffer[bufferHead]==0x2a || keyBuffer[bufferHead]==0x36){
11             keyBuffer[bufferHead]=0;
12             character-=0x20;
13         } //shift问题，先按shift，再按b，会被转化成B
14         注意大小写问题，将这个字符打印出来，然后和systemPrint类似

```

可以实现大写字母的打印和转换

```

Now I will test your getChar: 1 + 1 = 2
2 * 123 = 246
Now I will test your getStr: Alice is stronger than Bob
Bob is stronger than Alice
=====
Test end!!! Good luck!!!

```

## 3.8 实现 `getchar()` 函数

因为在 `keyBuffer` 里存储了输入的字符，所以说读取 `keybuffer` 即可实现，具体代码如下：

```

1  bufferHead = 0, bufferTail = 0; //init
2  keyBuffer[0]=0;
3  while(keyBuffer[bufferTail]!='\n') //遇到回车结束
4      enableInterrupt(); //允许中断
5  int character = keyBuffer[1]; //读取输入的第一个字符
6  tf->eax = character;
7  disableInterrupt(); //关闭中断

```

`getchar()` 自身代码如下：

```

1 | char character;
2 | character = syscall(SYS_READ, STD_IN, (uint32_t)&character, 1, 0, 0);
3 | return character;

```

## 3.9 实现getStr()函数

因为在keyBuffer里存储了输入的字符，所以说读取keybuffer即可实现，需注意因为要用到用户空间，所以需要进入用户态（模仿printf），具体代码如下：

```

1 |     bufferHead = 0;
2 |     bufferTail = 0;
3 |     keyBuffer[bufferHead] = 0;
4 |     int sel = USEL(SEG_UDATA);
5 |     asm volatile("movw %0, %%es"::"m"(sel));
6 |     char character = 0;
7 |     char* str = (char*)tf->edx;
8 |     while (keyBuffer[bufferTail] != '\n')
9 |         enableInterrupt();
10 |
11 |     for (int i = 0; i < 100; i++){
12 |         character = (char)(keyBuffer[i+1]);
13 |         if(character=='\n')break;//遇到回车结束
14 |         asm volatile("movb %0, %%es:(%1)"::"r"(character), "r"(str+i));
15 |     }
16 |     disableInterrupt();

```

getStr()自身代码如下：

```

1 | syscall(SYS_READ, STD_STR, (uint32_t)str, size, 0, 0);
2 | return;

```

## 四、实验介绍中的问题

**问题：**IA-32提供了4个特权级，但TSS中只有3个堆栈位置信息，分别用于ring0, ring1, ring2的堆栈切换。为什么TSS中没有ring3的堆栈信息？

**回答：**TSS是用来进行任务（进程）切换的，只有低特权级到高特权级切换时，新堆栈的信息才会从TSS中取得，ring3是最低特权级，故TSS中没有ring3的堆栈信息。

## 五、关于本次实验的感想

本次实验最大的问题就是不知道怎么bebug，用assert（0）也不会出现什么反应，我在irqHandle里遇到了一些问题，tf->irq是-1的时候我加了一个assert（0），但是此时应该不做任何操作即可。实验难度较上次有提升，但是框架代码给出的信息还蛮多的，整体实现比较有趣（感觉像在做数电实验一样），实验做完后也基本了解了printf和getchar以及getstr函数的实现，感觉收获还蛮多的。