

lab2 实验报告

姓名：刘佳璇

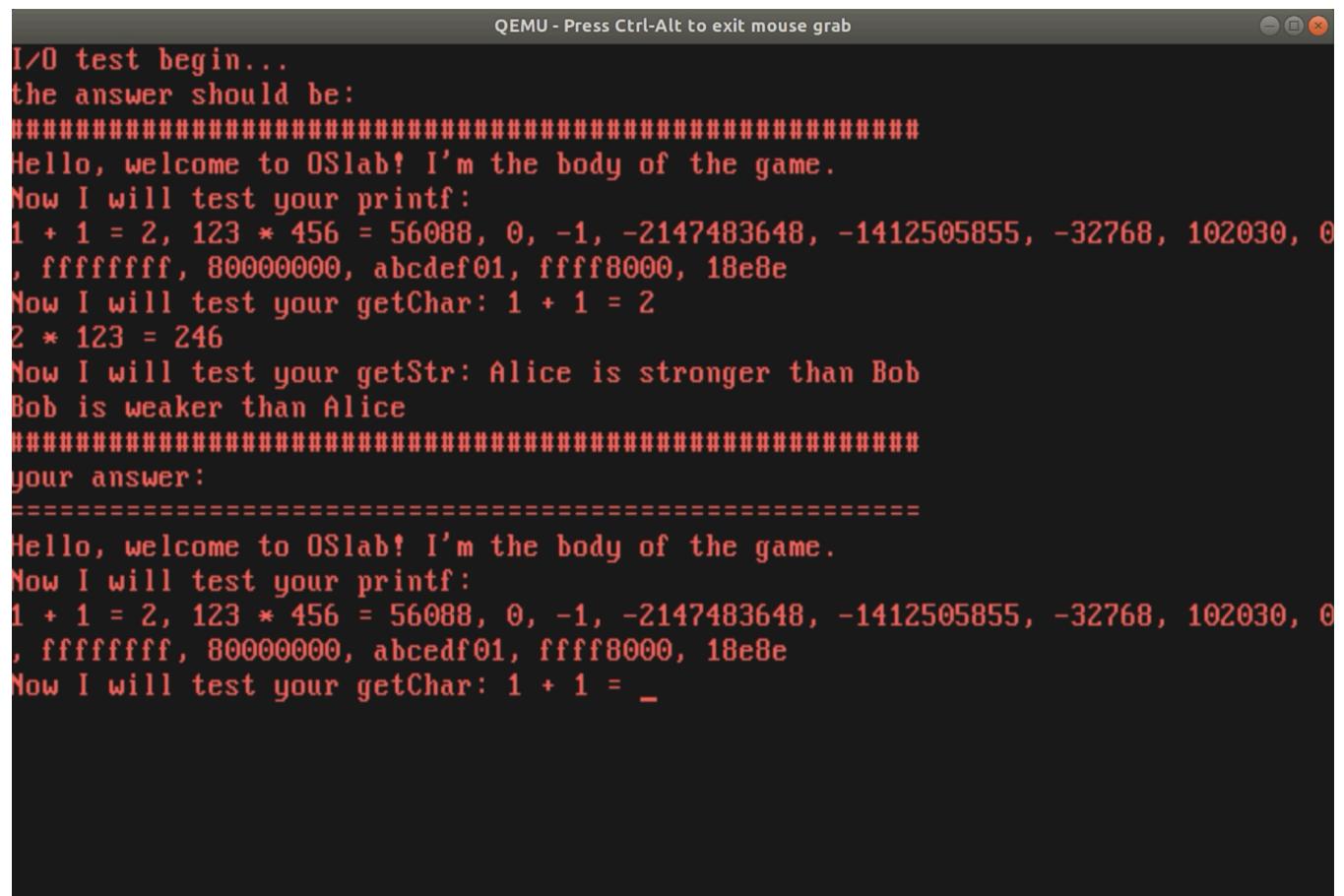
学号：231220105

邮箱：13573529579@163.com

实验进度

我完成了所有内容。

实验结果



The screenshot shows a terminal window titled "QEMU - Press Ctrl-Alt to exit mouse grab". The window contains the following text output:

```
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 = _
```

```
QEMU - Press Ctrl-Alt to exit mouse grab
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
, fffffff, 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
, fffffff, 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
=====
Sleep test begin...
```

```
QEMU - Press Ctrl-Alt to exit mouse grab
Bob is weaker than Alice
=====
Sleep test begin...
Current RTC time: 2025-3-29 8:35:58.
slept 1 second for 0 time(s).
Current RTC time: 2025-3-29 8:35:59.
slept 1 second for 1 time(s).
Current RTC time: 2025-3-29 8:36:0.
slept 1 second for 2 time(s).
Current RTC time: 2025-3-29 8:36:1.
slept 1 second for 3 time(s).
Current RTC time: 2025-3-29 8:36:2.
slept 1 second for 4 time(s).
Current RTC time: 2025-3-29 8:36:3.
slept 1 second for 5 time(s).
Current RTC time: 2025-3-29 8:36:4.
slept 1 second for 6 time(s).
Current RTC time: 2025-3-29 8:36:5.
slept 1 second for 7 time(s).
Current RTC time: 2025-3-29 8:36:6.
slept 1 second for 8 time(s).
Current RTC time: 2025-3-29 8:36:7.
slept 1 second for 9 time(s).
Test end!!! Good luck!!!
```

实验修改的代码位置

1. bootloader/boot.c

- bootMain()
 - line 20-25: 完善 bootMain 函数, 填写 kMainEntry、phoff、offset

2. kernel/kernel/idt.c

- setIntr()
 - line 16-23: 初始化中断门
- setTrap
 - line 29-36: 初始化陷阱门
- initIdt()
 - line 67-78: 初始化 IDT 表, 为中断设置中断处理函数

3. kernel/kernel/dolrq.S

- irqKeyboard
 - line 68: 将 irqKeyboard 的中断向量号压入栈

4. kernel/main.c

- kEntey()
 - line 12-28: 做一系列初始化

5. kernel/kernel/irqHandle.c

- 全局变量
 - line 18: 添加 lineBufferReady (行缓冲就绪标识)
 - line 20: 添加 timeFlag
 - line 36: 添加 sysGetTime() 的声明
 - line 401-409: 添加 struct TimeInfo
- irqHandle()
 - line 50-77: 填写中断处理程序的调用
- KeyboardHandle()
 - line 89-95: 初始化 lineBufferReady
 - line 99-108: 忽略断码、扩展码、修饰键
 - line 132-133: 设置行缓冲区就绪
 - line 146-171: 处理正常的字符
- timeHandler()
 - line 179: 设置 timeFlag = 1
- syscallHandle()
 - line 193-201: 添加跳转到 sysSetTimeFlag(), sysGetTimeFlag(), sysGetTime() 的 case
- sysPrint()
 - line 234-263: 完成光标的维护和打印到显存

- sysGetChar()
 - line 289-317: 实现 sysGetChar 函数, 使用行缓冲机制
- sysGetStr()
 - line 322-386: 实现 sysGetStr 函数, 使用行缓冲机制
- sysGetTimeFlag()
 - line 392: 实现 sysGetTimeFlag 函数
- sysSetTimeFlag()
 - line 398: 实现 sysSetTimeFlag 函数
- sysGetTime()
 - line 411-467: 实现 sysGetTime 函数

6. kernel/kernel/kvm.c

- loadUMain()
 - line 65-84: 完善 loadUMain 函数, 由 kernel 加载用户程序

7. lib/lib.h

- 全局
 - line 11-13: 添加 SYS_SET_TIME_FLAG, SYS_GET_TIME_FLAG, SYS_GET_TIME 的宏定义

8. lib/syscall.c

- getChar()
 - line 75-76: 实现 getChar 函数
- getStr()
 - line 82: 实现 getStr 函数
- printf()
 - line 105-182: 完善 printf 函数, 实现格式化输出
- sleep()
 - line 191-100: 实现 sleep 函数
- now()
 - line 206: 实现 now 函数, 使用 RTC 方式

实验思考

1. 在 kernel/kernel/kvm.c 中实现 loadUMain 时, 尝试过解析程序头表并逐个将节装载到内存的段, 装载后 app/main.c 不能如期运行。用 gdb 加载符号表并单步调试运行, 发现实际运行的汇编指令与反汇编文件中的指令不一致: 程序执行 `0x4 sub` 后应执行 `0x7 call`, 实际执行 `0x6` 处的指令。改为整体复制用户程序后恢复正常。
2. 在 kvm.c/enterUserSpace 的 `asm volatile("iret");` 前使用 `putStr()` 调试, 发现 `putStr()` 会更改栈顶, 导致返回值 `entry` 被更改。
3. 在 kernel/kernel/irqHandle.c 中填写 `irqHandle()` 中的中断处理程序时, 最初没有添加 `case -1: break;`, 导致程序启动过程中, 产生的未知系统中断跳转到 `default: assert(0);`, 使程序异常退出。

4. 在 kernel/kernel/irqHandle.c 中实现 KeyboardHandle() 时，最初没有处理 Shift 等修饰键，导致调用 getStr() 时 sysGetStr 读到 Shift 键就停止读入。
5. 在 kernel/kernel/irqHandle.c 中实现 sysGetStr() 时，最初没有过滤缓冲区的空白符，导致在 getChar() 或 getStr() 后调用 getStr() 时，sysGetStr() 读到上次键盘输入遗留在行缓冲区中的 '\n' 时就会停止读入，表现为不能正常读取字符。
6. 在 lib/syscall.c 中实现 now() 时，最初使用 outByte(), inByte() 函数从 RTC 中获取时间信息，但程序每运行到 outByte() 或 inByte() 就卡住。发现 now() 位于用户态，不能访问内核段后，改为采用系统调用实现 now()。