OSLab2 实验报告

姓名：凌嘉伟 学号：151220061

进度：

Lab2 基本全部完成

已实现一个简单的内核，将内核和游戏分开

已实现用户态格式化输出函数 printf：

仿照 printk 的测试，在游戏开始之前测试 printf：

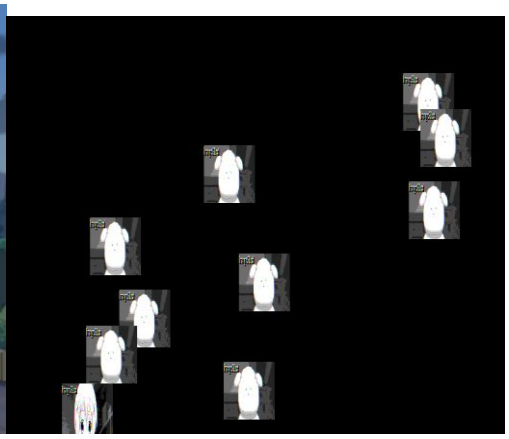
```
Start!
printf test begin...
the answer should be:
###################################################
Hello, welcome to OSlab! I'm the body of the game. Bootblock loads me to the mem
ory position of 0x100000, and Makefile also tells me that I'm at the location of
 0x100000. ~!@#$^&*()_+`1234567890-=...... Now I will test your printf: 1 + 1 =
2, 123 * 456 = 56088
0, -1, -2147483648, -1412505855, -32768, 102030
0, ffffffff, 80000000, abcdef01, ffff8000, 18e8e
###################################################
your answer:
===================================================
Hello, welcome to OSlab! I'm the body of the game. Bootblock loads me to the mem
ory position of 0x100000, and Makefile also tells me that I'm at the location of
 0x100000. ~!@#$^&*()_+`1234567890-=...... Now I will test your printf: 1 + 1 =
2, 123 * 456 = 56088
0, -1, -2147483648, -1412505855, -32768, 102030
0, ffffffff, 80000000, abcedf01, ffff8000, 18e8e
===================================================
```

已根据游戏要求实现系统调用：包括输出、时钟、键盘、画面

已将系统调用封装为库函数并将游戏重构

游戏：

由于本次实验要求，简单的字符界面的游戏无法适应要求，因此编写了一个射击游戏，键入 make qemu 之后，出现一个公园的背景，按下回车键，即可进入游戏（游戏画面失真）：



游戏的操作方法 :键盘 A 键-向左移动 ;键盘 D 键-向右移动 ;空格 space 键-发射子弹，子弹击中目标则目标消失，如果目标撞上玩家控制的人物，则游戏结束，按下回车键即可重新游戏。

思考题：

1、系统调用过程中如果涉及到指针的传送，由于段式存储的特点，我们需要知道指针对应的段到底是哪一个，这样我们才能获取对应的基地址。思考为什么扁平模式下我们不需要考虑这个问题。
Solution：扁平模式下，段起始地址为 0，段长度为地址空间最大长度，这样虚拟地址就和分段之后得到的线性地址一样了。这样可以“绕过”段式存储，避免这些问题。

2、为什么每个进程都需要自己的内核栈？
Solution：进入系统调用需要保存用户态运行时的寄存器信息，为了内核数据安全，用户修改内核数据需要从用户态切换为内核态。保存在低权限的用户态堆栈不安全，可能内核栈内存被用户态任意修改。

感想：

由于这次 oslab 的提交日期在清明节之后，因此尤其是这几天几乎每天都在调试 oslab，真的是放假一时爽，deadline…。在一开始的时候，由于自身能力有限，并不知道从何处下手，导致纠结苦思冥想了几天，甚至看到被调试得千疮百孔的代码，仍然不能完成跑出游戏，一度心灰意冷，想要放弃整个 lab。不过在指导之后，我也逐渐有了思路，重新写了 kernel 将内核和游戏分开，并且在此基础上完成了系统调用，重构了游戏，当看到游戏成功加载时，也逐渐有了信心。

PS：由于在分离游戏完成一部分系统调用之后，修改 Makefile 后不小心删去了 Makefile 中自动生成 git 记录的部分，因此 git 记录会有比较大的跳跃，之后查看才发现这一问题……_(:з ゝ ∠)_