L0: 直接运行在硬件上的小游戏(amgame)

171240511　孙旭东　匡亚明学院

一.游戏介绍

我实现了一个打地鼠小游戏，游戏的玩法如下：

1. 一共九个格子，白色代表安全，红色代表地鼠出现
2. 九个格子对应按键1～9, 顺序是

   ```
   1 2 3
   4 5 6
   7 8 9
   ```

   按下对应按键即可消灭该格地鼠，该格变为白色
3. 游戏的胜利条件是消灭全部30只地鼠
4. 游戏失败条件是按到了白色格子，或者所有格子都被地鼠占领

二.框架设计

1. 按顺序分别是处理玩家输入，生成地鼠，重新绘制图像
2. 处理玩家输入的频率是100HZ，生成地鼠的频率是3HZ，绘制图像的频率则是30HZ(FPS)

三.遇到的bug

因为设计的游戏比较简单，基本上没什么困惑的bug。印象比较深的一个，在单次调用draw_rect绘制图像时，对绘制的方阵大小是有限制的，我试图直接绘制160×160的方阵时会强退。

L1: 内核内存分配 (kalloc)

171240511　孙旭东　匡亚明学院

一.设计思路

1. 一开始想的是单向链表，从低到高(从左到右)找可分配空间，找到了就插入到尾部，free的时候和附近的一块(左或右)合并。
2. 把结构体数据放左边，那么free时只能往左合并。
3. 这样可以保证free之后的空间靠近head。
4. 但单向链表在free的时候要访问prev还得遍历，不值得，于是改成双向链表。
5. 此外，在链表结构体中放置uint32_t fence, 初始化为0xcccccccc，每次free前通过检查fence的数值判断要free的ptr是否合法。
6. 最后，因为对于上锁没有一点把握，选择了一把大锁锁住alloc，只要要分配内存，就给锁上。

二.遇到的bug

1. 麻烦在于没有测试样例，自己用循环搞测试样例时，发现并发输出的内容不太好设想。最后只能放弃全面并发的测试，稍微生成了一些锁住的样例，顺序测了下确认malloc空间里是我存进去的东西，而且free完后空间可以变回来。
2. 对于现在的L1没有自信，准备先杀到L2，不行再回来覆盖提交。
3. 有个bug，free的时候没有判地址有效性，所以free了错的地址，后面加了个结构体成员fence解决。
4. 从L2回来，感觉清晰多了，应该是没问题了。

L2: 内核多线程 (kthreads)

171240511　孙旭东　匡亚明学院

一.设计时想到的点

1. 信号量的睡眠与唤醒都要切换线程，所以irq中注册的切换函数必须要处理好这个。
2. 整个操作系统当成一个线程，编号0。(我一直想着写完了再加这个idle线程，结果因此后面出了bug)
3. sem_signal在中断中也可能被执行，所以yield的前要判断是否在中断中。(后来发现signal不用yield，因为唤醒之后不一定马上调度)
4. 多个处理器在中断处理时可能会去抢线程，给每个cpu弄一个线程调度链表。
5. 重点在于线程的调度，每次都把context存到当前线程的结构体上，然后要挑一个线程切换。

二.bug

1. 我线程调度是用链表实现，每次线程切换都把当前线程放到尾部，再把链表头部取出来。本以为没什么问题，但是sem信号量那里wait的时候也有个操作是把当前线程存到休眠队列，然后signal的时候再加到链表头部。这样就可能wait存完了，然后yield中断时又存了一遍，就导致该线程被加入头部两次，访问链表时就可能死循环。所以睡眠时要告诉中断处理这个线程别存了。
2. 信号量中，wait时不能带着锁yield，不然明明中断关着却进入中断。我选择yield前解锁，回来再上锁。但我的bug是没有线程了，默认的那两个线程睡眠着就没了。 搞定了，默认的两个线程是可以同时休眠的，所以自己必须弄一个idle死循环线程，我一开始想着idle没办法唤醒睡眠线程，没往这方面想，其实IO中断时input notify可以唤醒。
3. echo_task输出对的部分有的，但还有些不应该有的字符，发现是tty_write时用了sizeof，其实要用strlen。
4. echo_task一个没问题，开不只一个就出现AA型锁，ABBA锁，线程链表空了等各种问题。绝望之下放弃链表，改用数组，完全绑定线程和cpu(之前实现在wait和signal时线程可能换cpu)，每次handler遍历线程数组，wait和signal时就只要修改睡眠状态，而不用对线程链表做删除操作。 结果是2个echo能跑了，3,4个会出现AA型死锁(照xv6中加的panic)，怎么调都调不对，一气之下删了这句判断AA死锁的panic，结果居然能跑，那为什么报错?
5. 第4点中AA型死锁的问题找到了，我用的xv6的代码，它初始化lk->cpu=0,但它默认cpu从1开始编号，框架代码默认从0开始，所以lk->cpu在没上锁时应该置-1。