OSLAB1实验报告

余晨宁 151242062 匡亚明学院
151242062@smail.nju.edu.cn
2017.03.24

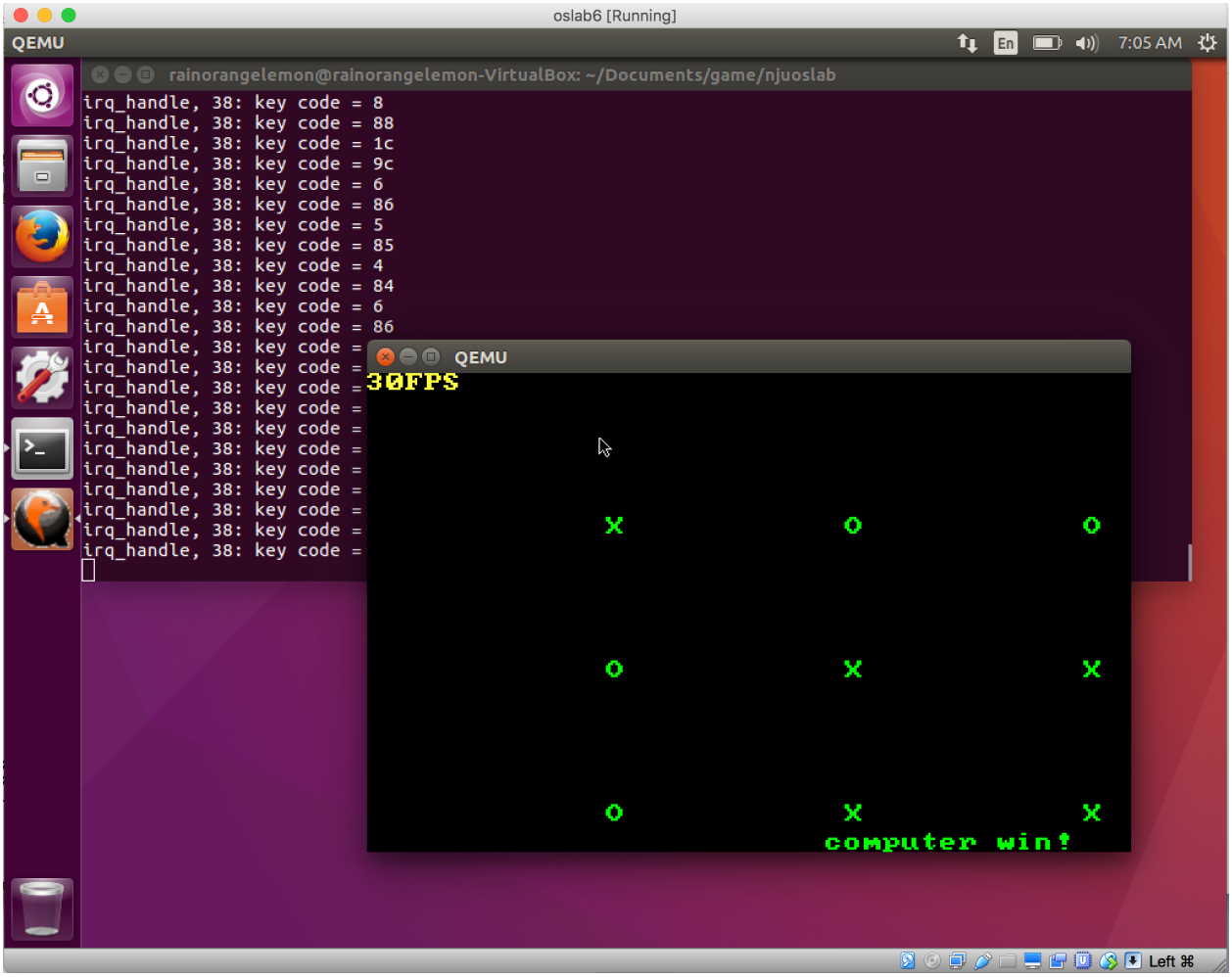
实验环境

ubuntu：64位16.04，gcc：5.3.1。

实验进度

我完成了框架代码的移植、printk以及中断的实现，并实现了一个简易的黑白棋。在游戏代码中保留了帧数的概念，有时间中断和键盘中断。

实验心得和想法

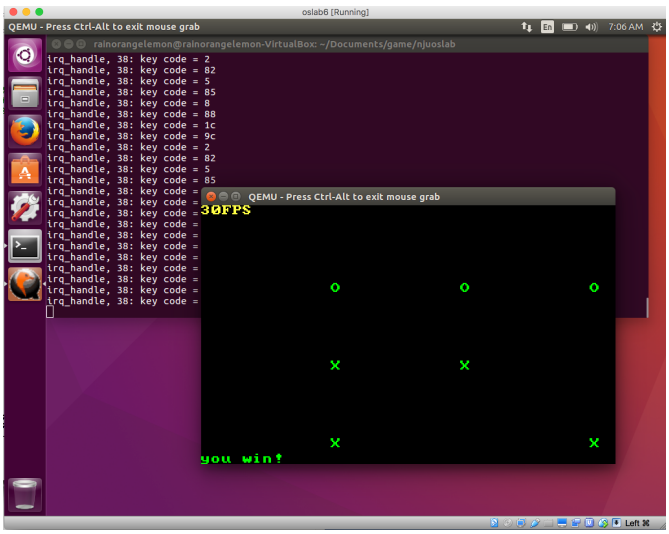


1、我实现的游戏的简要描述：

我实现的是一个九个方格的黑白棋，玩家可以通过输入1到9在合法的格子（即没有棋子已在其中的方格）下棋，如果有哪一方的三个棋子连成了一条线，那么屏幕就会显示出“computer win!”或者”you win!”。

缺点是没有记录胜负局的功能，输入enter能够刷新棋盘。

周围是几次棋局的截图：



2、一个玄学的改动：我做黑白棋实验的时候，发现输出的字符串尽管在draw.c中有写入，但是在屏幕更新的时候，它本来理应进行更新的，但是并没有成功。通过我的思考，我觉得框架代码进行了一定的优化，本来应该全部刷新的代码并没有得到全部的刷新，

而是只有局部的刷新。因此，通过查找包含display_buffer和prepare_buffer的源文件，我发现有个叫做PARTIAL_UPDATE的宏，定义了它之后就会有只重绘屏幕变化的部分。但这个变化的部分显然没有定义好，因此导致了我的部分变化的部分没有得到改变的bug。因此，我通过取消定义PARTIAL_UPDATE，使得整个的屏幕得到了刷新，从而得到了我想要的结果。

3、游戏上的一些心得：徐老师在操作系统课上说，实现微内核结构的核心思想是要尽量减少mechanism和policy的耦合程度。我觉得即使对于游戏代码，它的mechanism和policy也是要尽量减少耦合的。比如说device文件夹下面的提供的就是mechanism，在game文件夹下面，相比较于draw.c和keyboard.c，game.c和effect.c是提供policy的，负责描述游戏的运行逻辑，而draw.c和keyboard.c则负责游戏和硬件的交互。这样写的一个好处是当出了bug的时候，容易找到出错的是哪一个部分，不需要一个文件一个文件地去找。

4、整个心路历程：我在做《计算机系统基础》的实验的时候用的是docker+XQuartz的模式，但是在这次实验的时候，虽然实验讲义中有说可以使用窗口转发的方法，并且说详见pa讲义，但是我就是按照pa实验讲义做的，依然搞不出来qemu的转发窗口。后来我看群里面卢以宁说要装GUI，我百度了一下，发现docker里装GUI似乎属于前沿科技，而且我也找不到具体的方法（这说明我的搜索能力有待提高）。最后我放弃了，转而使用virtualbox。一开始装ubuntu16.04的32位iso，结果装完之后第二次开系统的时候鼠标和键盘就无法响应了。我想搞清楚是这是偶然现象还是必然现象，于是我又新装了一个容器。然后我又碰到了这样的问题。我上网搜了一下，说是要在terminal里面输上一些命令然后就能解决无法响应的问题了。但是为了解决这个问题，前提是键盘和鼠标中必须有一个是能够响应的。但是我两个都无法响应，因此我放弃了。然后我尝试装了一个64位的系统，想看看这个问题是我的镜像有损坏导致的还是我的virtualbox有bug导致的。结果鼠标和键盘能够响应，我觉得这是我的镜像损坏导致的。后来在有图形界面的virtualbox的虚拟机里，qemu自然而然的能够弹出窗口了。这告诉我们，简单为美。virtualbox的使用很简单，而转发窗口要经过的路径很复杂，很难找出是其中哪个地方出了差错。另外不要用学校网并且用chrome下大的文件（比如1.5G的iso），有可能会损坏。

5、少用全局变量。在原本的游戏代码中，我讶异于hit和miss的引用方法：直接把miss和hit作为int型的全局变量不就行了吗？但是事实并非如此，在之后的写游戏过程中，我意识到如果变成全局变量，可能这两个全局变量会遭到多处代码的修改，并且指不定就有命名上的重复，有可能导致代码的不可预测。

总得来说，此次虽然遭受了环境的大转移，从docker转移到了virtualbox，但是还是挺幸运的，没有什么难以理解的bug。

此外还请助教不要奇怪于我的git log，因为我之前一直在准备GRE，上周六才刚考完，之后又补了一大堆作业，因此这周才开始写，以后我一定会注意均匀地分配好时间的。