Lab2 实验报告

史子凡　151180115　903498500@qq.com

一、实验进度：

我已完成所有内容。采用了分页机制的实现。游戏成功运行。

二、GCC version：4.8.4 debian

三、讲义中的问题：

1、

```
Q：如果你参考jos的代码，在entry.S中，有这样2行代码：
_start = RELOC(entry)
...
jmp *%eax
请分别解释这2行代码的意义
```

_start = RELOC(entry)：

entry.S 在运行时其实是被链接到了虚拟地址（在 kernel.ld 中完成），是一个高地址，所以 entry 在 0xc0100000 的位置，但是刚开始进入 kernel 的时候我们还没有进行分页操作，所以虚拟地址是不可能被访问到的，而 _start = RELOC(entry)就将 entry 的值减去 KERNBASE 赋给_start，也就是代码段入口的物理地址，这样就可以正常访问了。

jmp *%eax

这句的作用就是进行虚拟地址的跳转。地址在 eax 中然后使用 jmp 跳转到虚拟地址开始执行。本质上是对 eip 指针的变动。

2、

首先，因为进程后面需要进行系统调用，系统调用需要执行kernel中的代码，所以你需要将kernel的页目录拷贝一份作为进程的页目录的模板。

```
Q：这样做为什么可以，会不会带来什么问题？
```

这样做是可以的，因为这样用户就能够访问到 kernel 中的东西了，比如需要系统调用的代码之类的。

但也正是因为这样而带来了问题，因为用户可以访问到 kernel 中的东西从而有可能去篡改代码或者数据，实施攻击，很危险。

四、遇到的问题:

1. 刚开始时真的是一脸懵逼，完全不知道怎么下手，后来参考了 PA 还有大班的 oslab 的框架，总算明白了是怎么一回事了，有点觉得 PA 其实没学什么。

还有就是 JOS 的框架代码我感觉还是挺难读的，自己对分页机制其实了解的没有那么好。后来到网上找了一个解读 JOS 框架代码的博客，才少走了很多弯路。

2. 分离 kernel 和 game 的时候在头文件上搞了好久，还是有各种大大小小的问题，主要是依赖关系的问题。后来听取别人的建议，像 PA 那样，把 kernel 和 game 的头文件分开来，分别放在 kernel 和 game 的文件夹中，这样头文件分开来处理起来果真就简单了许多。

3. 最恶心的大概就是各种缺页的 bug 了。不过大多数是因为自己对分页机制并不是特别清楚还有….真的不太细心。比如从用户程序的中断进入 kernel 处理的时候，才开始忘了给(KSTACKTOP-KSTKSIZE)~KSTACKTOP 这一段分配物理空间（tss 的 esp 设置的是 KSTACKTOP），所以一进 kernel 就开始缺页。