Lab3 实验报告

史子凡　151180115　903498500@qq.com

一、 实验进度：

我已完成了要求完成的内容，实现了 fork,getpid,exit,sleep 四个系统调用和进程切换等。

二、 GCC version：4.8.4 debian

三、 遇到的问题和感想。

1. 同样这次实验开始时还是有点蒙蔽的，然后参考了 12 级的讲义以及框架代码。于是就采取了循环队列这种从来没有接触过的链表方式来管理 ready,block,free 三个队列。所幸的是 12 级的框架代码里给了一些关于双向链表的接口，好好学习了一波双向链表的管理以及一些没见过的宏定义。比如：

```
#define list_entry(ptr, type, member) \
    ((type*)((char*)(ptr) - (int)(&((type*)0)->member)))
```

这个里面出现的 0，才开始一点也不能理解，后来上网搜了好一波资料才找到解释，这里是将 0 强制转化为 type 类型的指针，并且取出 member 变量的地址，也就是得到 member 所指向的变量相对于整个结构体变量的相对地址。对于这个 list_entry 这个宏，开始也是不能理解的，后来才明白它就是找到 list 对应的 pcb 的。

2. pcb 的结构我也借鉴了 12 级讲义里提到的用 union 来管理的方法，但是遇到了一点麻烦。

```
typedef union ProcessControlBlock {
    uint8_t kstack[KSTACK_SIZE];
    struct{
        TrapFrame *tf;
        pde_t *pgdir;
        list plist;
        uint32_t pid,ppid;
        int state;
        int timecount;
        int sleeptime;
        void *addr;
    };
    //struct ProcessControlBlock *pcbo;
}PCB;
```

因为 union 里面是共享地址空间的，所以关于进程信息的 struct 和 kstack 是连在一起的，在 fork 进行深拷贝的时候，才开始没有注意到这点，然后拷贝了整个 kstack，于是进程的信息就全被覆盖掉了。后来又读了遍 12 级的讲义，才发现了这个梗。于是就在 struct 里加上一个指针，专门用来得到 struct 的偏移地址，可以像在上面 list 中提到的方式得到相对地址，然后复制的时候特殊处理就可以了。

3. 这次 de 的最久的但也最弱智的一个 bug 是缺页错误。最开始通过查看寄存器的值发现是每次 int 80 回来后的那个语句崩了，然后就以为自己的修改中

断处理的时候写错了，结果看了好久也没看出来。然后将调用的系统调用好打出来，逐渐缩小了出错的系统调用的范围，最终锁定在了 fork 上。深拷贝乍一看没有错，后来把每一步的值都输出来才发现是因为在深拷贝时，page directory entry address 的掩码写错了，写成了和 page table 一样了。

4. 还遇到一些奇怪的问题，比如刚写完时运行游戏，结果 qemu 的页面忽大忽小，不断的 initialize。后来发现是 pmap 里面有地方写错了。

5. 在进程结束的时候，对 pcb 进行了一定的回收，`pcb_remove(pcbnow);`（在 pcb 中），回收了分配的页面，具体回收页面写在了 pmap.c 里面。

6. 在 game.c 里面写了系统调用的代码后，游戏运行变慢了，不过系统调用都是可以正常运转的。