操作系统——Lab5

学号：2013921
姓名：周延霖
专业：信息安全

> 由于没有什么抱怨的，所以再次回答一下指导书上提出的问题

练习1：加载应用程序并执行（需要编码）

> 然后就是描述当创建一个用户态进程并加载了应用程序后，CPU是如何让这个应用程序最终在用户态执行起来的过程:

1. 使用mm_create来申请一个新的mm并初始化
2. 使用setup_pgdir来申请一个页目录表所需的一个页大小，并且把ucore内核的虚拟空间所映射的内核页表boot_pgdir拷贝过来，然后mm->pgdir指向这个新的页目录表
3. 根据程序的起始位置来解析此程序，使用mm_map为可执行程序的代码段，数据段，BSS段等建立对应的vma结构，插入到mm中，把这些作为用户进程的合法的虚拟地址空间
4. 根据各个段大小来分配物理内存，确定虚拟地址，在页表中建立起虚实的映射。然后把内容拷贝到内核虚拟地址中
5. 为用户进程设置用户栈，建立用户栈的vma结构。并且要求用户栈在分配给用户虚空间的顶端，占据256个页，再为此分配物理内存和建立映射
6. 将mm->pgdir赋值给cr3以更新用户进程的虚拟内存空间。
7. 清空进程中断帧后，重新设置进程中断帧以使得在执行中断返回指令iret后让CPU跳转到Ring3，回到用户态内存空间，并跳到用户进程的第一条指令。

需要注意的是，在第六步的时候，init已经被exit所覆盖，构成了第一个用户进程的雏形。在之后才建立这个用户进程的执行现场

- `do_execve`函数部分执行用户进程的创建工作
- `load_icode`函数来给用户进程建立一个能够让用户进程正常运行的用户程序
- 用户进程的用户环境已经搭建完毕。此时initproc将按产生系统调用的函数调用路径原路返回，执行中断返回指令“iret”后，将切换到用户进程的第一条语句位置_start处开始执行

练习2：父进程复制自己的内存空间给子进程（需要编码）

> 请在实验报告中简要说明如何设计实现”Copy on Write 机制“，给出概要设计，鼓励给出详细设计。
>
> Copy-on-write（简称COW）的基本概念是指如果有多个使用者对一个资源A（比如内存块）进行读操作，则每个使用者只需获得一个指向同一个资源A的指针，就可以该资源 了。若某使用者需要对这个资源A进行写操作，系统会对该资源进行拷贝操作，从而使得 该“写操作”使用者获得一个该资源A的“私有”拷贝—资源B，可对资源B进行写操作。该“写 操作”使用者对资源B的改变对于其他的使用者而言是不可见的，因为其他使用者看到的 还是资源A。

- `do_fork`：进行内存复制时，如在copy_range中，不进行复制，而是将父进程和子进程的虚拟页映射上同一个物理页，然后将父进程的PDE直接赋值给子进程，将PTE_W置为0(不可写入)，但应用程序试图写入某个共享页就会产生页访问异常

- `page_fault`：在page_fault的ISR部分，增加对这种异常的处理，处理方法为将当前的共享页的内容复制过去，建立出错的线性地址与新创建的物理页面的映射关系，将PTE设置设置成非共享的；然后查询原先共享的物理页面是否还是由多个其他进程共享使用的，如果不是的话，就将对应的虚地址的PTE进行修改，删掉共享标记，恢复写标记；

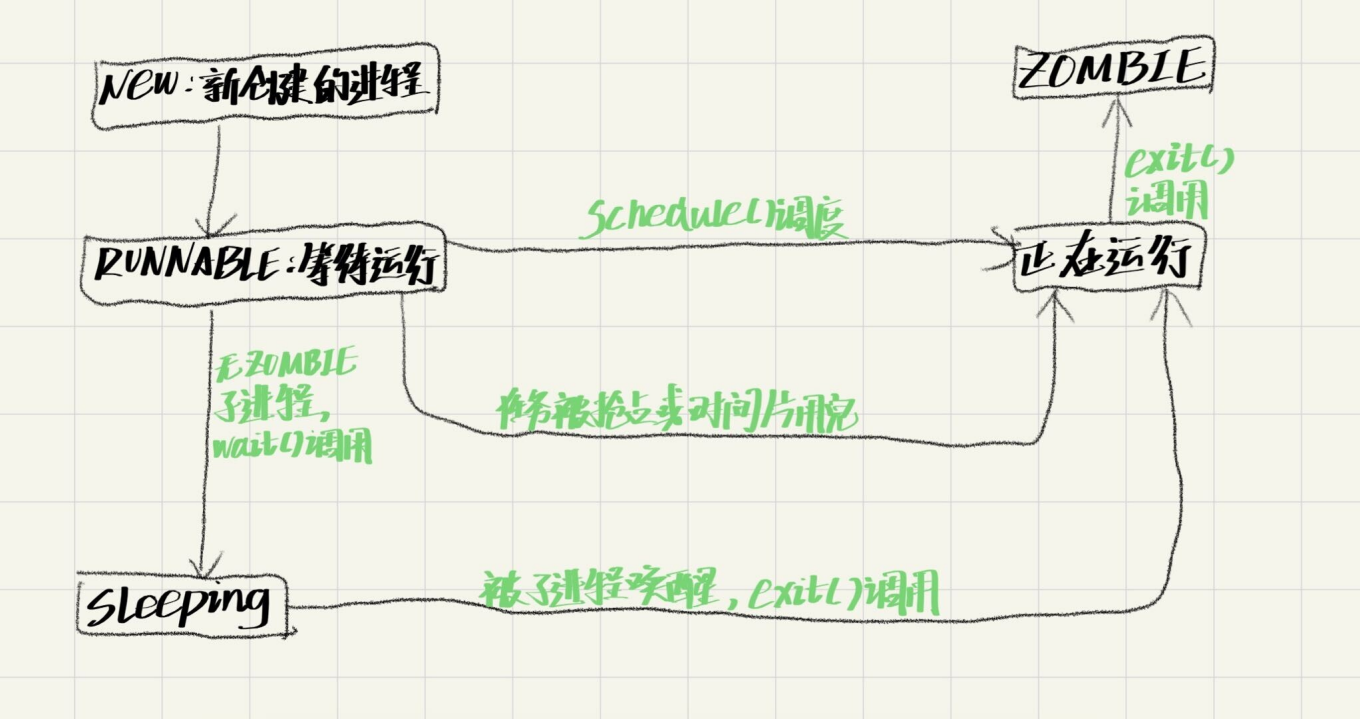
练习3：阅读分析源代码，理解进程执行 fork/exec/wait/exit 的实现，以及系统调用的实现（不需要编码）

> 请在实验报告中简要说明你对 fork/exec/wait/exit函数的分析。并回答如下问题：
>
> - 请分析fork/exec/wait/exit在实现中是如何影响进程的执行状态的?
> - 请给出ucore中一个用户态进程的执行状态生命周期图（包执行状态，执行状态之间的变换关系，以及产生变换的事件或函数调用）。（字符方式画即可）

1. 请分析fork/exec/wait/exit在实现中是如何影响进程的执行状态的

    - `fork`：不会影响当前进程的执行状态，但是会将子进程的状态标记为RUNNALB，使其可以在后续的调度中运行
    - `exit`：不会影响当前进程的执行状态，但是会修改当前进程中执行的程序
    - `execve`：系统调用取决于是否存在可以释放资源（ZOMBIE）的子进程，如果有的话不会发生状态的改变，如果没有的话会将当前进程置为SLEEPING态，等待执行了exit的子进程将其唤醒
    - `wait`：将当前进程的状态修改为ZOMBIE态，并且会将父进程唤醒，对该进程的资源进行回收

2. 请给出ucore中一个用户态进程的执行状态生命周期图（包执行状态，执行状态之间的变换关系，以及产生变换的事件或函数调用）



总结

本次实验主要涉及进程的一些进程的知识，比如创建，管理，切换到用户态进程的具体实现；加载ELF可执行文件的具体实现；对系统调用机制的具体实现

理论课老师就讲过进程管理是操作系统很重要的一个知识点，也特别提到了进程五状态，这一点也在实验中有体现。challenge实现的COW，在Linux上也有体现所以相关资料很多，讲的十分详细，对challenge的实现有很大帮助，而且challenge也有注释提示

做完这次实验，对进程管理有了更深的认识，同时对理论课上学到的知识有了更进一步的理解，还能查漏补缺，发现自己遗忘或者没学好的知识点。期待自己未来更好的发展，万事胜意、心想事成、未来可期