OS Lab3

520030910393 马逸川

练习题1:

内核从完成必要的初始化到用户态程序的过程是怎么样的？尝试描述一下调用关系。

参考`main.c`代码，在这一过程中，首先调用`create_root_thread();`创建线程，`eret_to_thread`切换到选中的线程。

练习题2:

练习2及以下的练习部分的实现参考了这一篇博客的内容。

练习2各项实现思路:

cap_group_init：

参考文档中给出的cap_group拥有的四项参数。将四个参数分别用对应的函数初始化即可。

sys_create_cap_group&create_root_cap_group：

参考文档中给出的提示内容，调用obj_alloc分配cap_group和vmspace对象，并通过cap_group_init函数初始化对应的参数。函数具体调用时填入的参数参考了博客的实现，两函数的实现思路相同。

###

练习题3:

load_binary:

这一部分的实现参考了博客的讲解:

分配内存时，将 `p_memsz` 转换成页对齐的 `seg_map_sz`。`p_memsz` 开头要扩展到前一个页的交界处，结尾要扩展到后一个页的交界处。按页对齐是为了保证 `p_vddr` 在虚拟页里的便宜和物理页里的便宜相等，进而提高寻址速度。拷贝操作使用了memcpy函数。

练习题4:

参考博客，填写异常向量表即可。

配置的两个跳转都是需要返回unexpected_handler的异常，用bl操作符跳转。

练习题5:

调用handle_trans_fault函数。

练习题6:

按照所给的提示，如果get_page_from_pmo(pmo, index)=0，意味着物理页未分配，调用函数将其分配并记录在pmo中添加映射，否则直接添加映射。

练习题7:

在`arch/machine/registers.h`中查询对应的偏置。

先预留所有寄存器的大小，按顺序保存需要使用的寄存器。

在退出异常中断时，从预留的位置读取保存的寄存器值即可。

练习题8:

输入输出字符的部分使用uart.h中定义的函数即可。

下面阐述三个系统调用:

sys_thread_exit： 设置退出状态并将thread置为null。

__chcore_sys_putc：参考syscall_arch.h文件中的定义和syscall_num.h文件中定义的参数，按照需要的功能调用chcore_syscall1函数即可。

如putc需要调用两个参数：__CHCORE_SYS_putc和需要被输出的参数，所以调用chcore_syscall1，也即有两个参数输入的系统调用。

getc需要一个参数，所以只需输入__CHCORE_SYS_getc参数即可。

__chcore_sys_thread_exit同样，只需要输入CHCORE_SYS_thread_exit完成系统调用即可。

挑战题9:

(这部分的实现copy了博客的内容，下面说的是我的理解)

对所有的page，维护一个bool数组is_mapped来标记被换出还是换入，如是，则进行换出操作，使用memcpy，并调用 unmap_range_in_pgtbl在页表中删除此项。

若布尔值为false，则进行换入操作，在页表中分配此项即可。