lab3：进程与线程，异常处理

> author: 魏新鹏
>
> student ID：519021910888

1　练习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。

在 main.c 中，先调用 arch_interrupt_init 初始化异常向量表，然后调用 create_root_thread，它会先创建 root cap_group，然后在这个group中创建 root thread，并将 current_thread 切换为 root_thread，然后调用 switch_context 切换页表，并找到 root_thread 的TCB（thread control block）将其作为参数传给 eret_to_thread，该函数切换栈将TCB的地址切换为新的栈指针，调用 exception_exit 恢复通用寄存器的值以及三个特殊寄存器（ sp_el0 , elr_el0 , spsr_el1 ）的值并最终调用 eret 返回。

1.1　创建完cap_group（process）之后是如何进一步创建root thread的?

1. 分配栈空间：start: 0x500000000000 size: 0x800000。由于栈是从高往低走的，所以是从0x50000800000到0x50000000000。
2. load binary把elf的各个段load进来。
3. 然后给栈分配了一个内存页用来prepare env（放metadata啥的），因此 __init_chcorelibc 中sp不是0x50000080000而是0x5000007ff000

   > 这就是为什么page fault的时候会出现pa不等于0（有物理页）但是没页表（因为此时是在内核态，用户态的页表还是没的）的情况）init_chcorelibc会先读sp处的一个long，而此时sp指向这个物理页的底部，读的这个long是在这个物理页里的

4. thread init。主要初始化了tcb（struct thread保存了thread_tcx的指针，thread_tcx通过create_thread_tcx初始化，这个函数会精细控制tcb的位置，详见（kernel/sched/context.c））
5. 把这个thread list_add到cap_group的thread list中。