1. trap\_frame与context的区别是什么？
   * 从内容上看，trap\_frame包含了context的信息，除此之外，trap\_frame还保存有段寄存器、中断号、错误码err和状态寄存器eflags等信息。
   * 从作用时机来看，context主要用于进程切换时保存进程上下文，trap\_frame主要用于发生中断或异常时保存进程状态。
   * 当进程进行系统调用或发生中断时，会发生特权级转换，这时也会切换栈，因此需要保存栈信息（包括ss和esp）到trap\_frame，但不需要更新context。
2. trap\_frame与context在创建进程时所起的作用：
   * 当创建一个新进程时，我们先分配一个进程控制块proc，并设置好其中的tf及context变量；
   * 然后，当调度器schedule调度到该进程时，首先进行上下文切换，这里关键的两个上下文信息是context.eip和context.esp，前者提供新进程的起始入口，后者保存新进程的trap\_frame地址。
   * 上下文切换完毕后，CPU会跳转到新进程的起始入口。在新进程的起始入口中，根据trap\_frame信息设置通用寄存器和段寄存器的值，并执行真正的处理函数。可见，tf与context共同用于进程的状态保存与恢复。
   * 综上，由上下文切换到执行新进程的处理函数fn，中间经历了多次函数调用：forkret() -> forkrets(current->tf) -> \_\_trapret -> kernel\_thread\_entry -> init\_main.