实现功能：

1. 在 TaskManagerInner 结构体（保存每个进程的信息）中加入 task_info:TaskInfo2 保存每个进程的状态，系统调用次数和开始运行的时间，TaskInfo2 和自带的 TaskInfo 的区别在于，由于目前只有 5 个系统调用，因此以长度为 5 的数组存储系统调用次数。
2. 在第一次运行进程时，自动更新 task_info 中的开始运行时间和进程状态。
3. 加入函数 add_cur_syscall_times，根据系统调用编号增加当前进程的系统调用次数。
4. 在系统调用入口 syscall 函数调用 add_cur_syscall_times，统计系统调用次数。
5. 加入要求实现的系统调用 sys_task_info，会读取 task_info 中信息并存入 ti 指针指向的地址。

问答题：

1.

ch2b_bad_address.rs：访问无效地址 0x0 报错
ch2b_bad_instruction.rs：在 U 态使用 S 态特权指令 sret 报错
ch2b_badregister.rs：在 U 态访问 S 态寄存器 sstatus 报错
sbi：RustSBI version 0.3.0-alpha.2, adapting to RISC-V SBI v1.0.0

2.

1.a0 为 trap_handler 的返回值（cx：TrapContext）。__restore 的使用场景有：从系统调用中返回，加载用户程序上下文并将控制转给用户程序。
2.特殊处理了 sstatus,sepc 和 sscratch 寄存器。sstatus 保存了陷入前的处理器特权级别等信息，sepc 保存了陷入的指令位置（即需要返回到的地址），sscratch 为保存的用户栈的栈指针。
3.x2 为栈指针，需要保存在 sscratch 寄存器中而不是栈中。x4 为线程指针寄存器，由于我们目前只使用单线程，无需使用该寄存器，因此不保存。
4.sp 为用户栈指针，sscratch 为内核栈指针。
5.sret 指令。由于 sret 指令会恢复 sstatus 寄存器中的 SPP 位保存的陷入前的特权级别，陷入前处于 U 态，因此会返回 U 态。
6.sp 为内核栈指针，sscratch 为用户栈指针。
7.在进行系统调用时执行的 ecall 指令。

荣誉准则

在完成本次实验的过程（含此前学习的过程）中，我曾分别与 以下各位 就（与本次实验相关的）以下方面做过交流，还在代码中对应的位置以注释形式记录了具体的交流对象及内容：

无

此外，我也参考了 以下资料 ，还在代码中对应的位置以注释形式记录了具体的参考来源及内容：

无

3. 我独立完成了本次实验除以上方面之外的所有工作，包括代码与文档。 我清楚地知道，

从以上方面获得的信息在一定程度上降低了实验难度，可能会影响起评分。

4. 我从未使用过他人的代码，不管是原封不动地复制，还是经过了某些等价转换。 我未曾也不会向他人（含此后各届同学）复制或公开我的实验代码，我有义务妥善保管好它们。 我提交至本实验的评测系统的代码，均无意于破坏或妨碍任何计算机系统的正常运转。 我清楚地知道，以上情况均为本课程纪律所禁止，若违反，对应的实验成绩将按“-100”分计。