os lab1实验报告

计算机系 2019011312 姚建竹

编程作业

1. 在TaskManager中设置一个函数，可以返回当前正在运行当中的task的TaskControlBlock
2. 在block中加入信息：syscall_list和start_time
3. 在TaskManager的run_first_task和run_next_task内，获得不同task的start_time
4. 在syscall/mod.rs的函数里加上一个增加update_syscall_items的函数，将syscall_id传回，设置对应task的syscall_times数组

简答题

正确进入 U 态后，程序的特征还应有：使用 S 态特权指令，访问 S 态寄存器后会报错。 请同学们可以自行测试这些内容 (运行 Rust 三个 bad 测例 (ch2b_bad_*.rs) ， 注意在编译时至少需要指定 LOG=ERROR 才能观察到内核的报错信息) ， 描述程序出错行为，同时注意注明你使用的 sbi 及其版本。

[rustsbi] RustSBI version 0.2.0-alpha.4

- ch2b_bad_address.rs：[ERROR] [kernel] PageFault in application, bad addr = 0x0, bad instruction = 0x8040008a, core dumped.
- ch2b_bad_instructions.rs：[ERROR] [kernel] IllegalInstruction in application, core dumped.
- ch2b_bad_register.rs：[ERROR] [kernel] IllegalInstruction in application, core dumped.

深入理解 trap.S 中两个函数 __alltraps 和 __restore 的作用，并回答如下问题:

1. 刚进入__restore时，a0代表内核栈栈顶；刚开始运行某程序的时候可以使用__restore，返回trap要恢复上下文时使用__restore
2. sstatus, sepc, sscratch，作用分别为：指出Trap发生之前CPU 处在哪个特权级（S/U）等信息（如果之前是用户态，那么就是U）；当 Trap 是一个异常的时候，记录 Trap 发生之前执行的最后一条指令的地址，这样可以正确返回到指定的用户态代码；而sscratch保存了用户栈顶，需要恢复
3. sp(x2)的话是因为已经指向内核栈了，而x4是因为应用不会使用到这个寄存器

4. 做用户栈和内核栈的栈顶交换，从而恢复用户栈顶为sp，sscratch也变回了内核栈顶。
5. sret。因为sret指令让 CPU 将当前的特权级按照 sstatus 的 SPP 字段进行设置，然后跳转到 sepc 指向的位置。由于此时 SPP 为 User，因此会进入用户态。
6. 做用户栈和内核栈的栈顶交换，从而sccratch保存了用户栈顶，sp也切换到了内核栈顶。
7. 发生中断异常的那条指令。