rCore ch3 lab repoter

Contents

1. 实现功能

通过修改 `TaskControlBlock` 结构体，增加对于每个任务而言的系统调用记录数组 `syscall_times`，该任务的开始时间 `task_start_time` 以及该任务是否被处理过 `processed`。添加了额外的 `TaskManager` 的实现，一个来获取该任务的 `taskControlBlock` 信息，一个则可以在访时 `syscall` 时使该任务的系统调用记录增加 1。 对于任务首次处理的时间可以在 `run_next_app` 函数中通过对 `processed` 对判断来记录。

至此，实现了系统调用 `sys_task_info`。

2. 问答题

2.1. 第一题

以 `qemu-riscv64` 虚拟环境分别执行 `ch2b_bad_register.elf`, `ch2b_bad_instructions.elf`, `ch2b_bad_address.elf`输出如下

```
root@ubuntu:~/rCore/user/build/elf# qemu-riscv64 ch2b_bad_register.elf
Illegal instruction

root@ubuntu:~/rCore/user/build/elf# qemu-riscv64 ch2b_bad_address.elf
Segmentation fault

root@ubuntu:~/rCore/user/build/elf# qemu-riscv64 ch2b_bad_instructions.elf
Illegal instruction
```

SBI 为 `[rustsbi] Implementation: RustSBI-QEMU Version 0.2.0-alpha.2`

2.2. 第二题

1. 当内核第一次执行用户程序时，`a0` 存储的是手动初始化的 Trap Context 的压入内核栈后的栈顶指针，当内核在 U 特权级和 S 特权级之间切换的时候，`a0` 存储的是要切换到的那个程序的内核栈中的 Trap Context 的栈指针，所以 `__restore` 有两个使用场景：一是在 U/S 特权级之间切换，二是首次运行程序。
2. 从内核栈中读取并保存到 `t0, t1, t2` 寄存器的值分别是其要进行返回的程序的 `sstatus, spec, sscratch` 三个 CSR 寄存器的值，`sstatus` 的 `SPP` 字段存储 CPU 当前的特权级(U/S)。`sepc`

存储 Trap 处理完成后默认会执行的下一条指令的地址。而 `sscratch` 则作为一个临时寄存器存储将要返回的任务的用户栈。

3. 在执行 `__alltraps`时： `tp(x4)` 寄存器，除 非我们手动出于一些特殊用途使用它，否则一般也不会被用到。我们在这 里也不保存 `sp(x2)`，因为我们要基于它来找到每个寄存器应该被保存到的正确的位置
4. 两个寄存器的值发生了交换，此时 `sp` 存储指向用户栈的指针，而 `sscratch` 存储只想内核栈的指针。
5. `sret` 指令，其是 Risc-v 模式特权指令，可以从 S 模式，返回到 U 模式。
6. 两者发生交换，此时 `sp` 存储指向内核栈的指针，而 `sscratch` 存储指向用户栈的指针。
7. `call trap_handler`

3. 荣誉守则

1. 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。
2. 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。