

# ch3实验报告

## 实验

实现获取当前进程状态可以给 `task/mod.rs` 的 `TaskManager` 结构实现一个 `get_current_status` 的方法读取自身私有字段 `status`, 然后外面写一个函数把给全局变量 `TASKMANAGER` 做一个反射就行

实现记录 `syscall` 调用次数可以给 `TaskControlBlock` 结构体加上一个 `[u32; MAX_SYSCALL_NUM]` 的成员负责存储 `syscall` 调用次数记录, 在 `syscall/mod.rs` 里面加上记录 `syscall` 调用次数的函数, 把 `syscall_id` 作为参数传进去

实现获取进程运行时间可以给 `TaskControlBlock` 结构体加上一个进程启动时间的参数, 获取进程运行时间就调用 `get_time_ms()` 减去进程启动时间得到进程运行时间

## 简答题

1

运行截图

```
[rustsbi] RustSBI version 0.3.0-alpha.2, adapting to RISC-V SBI v1.0.0
[rustsbi]
[rustsbi] Implementation : RustSBI-QEMU Version 0.2.0-alpha.2
[rustsbi] Platform Name   : riscv-virtio,qemu
[rustsbi] Platform SMP    : 1
[rustsbi] Platform Memory : 0x80000000..0x88000000
[rustsbi] Boot HART       : 0
[rustsbi] Device Tree Region : 0x87000000..0x87000ef2
[rustsbi] Firmware Address  : 0x80000000
[rustsbi] Supervisor Address : 0x80200000
[rustsbi] pmp01: 0x00000000..0x80000000 (-wr)
[rustsbi] pmp02: 0x80000000..0x80200000 (---)
[rustsbi] pmp03: 0x80200000..0x88000000 (xwr)
[rustsbi] pmp04: 0x88000000..0x00000000 (-wr)
[kernel] Hello, world!
[ERROR] [kernel] .bss [0x80263000, 0x8028c000)
[kernel] PageFault in application, bad addr = 0x0, bad instruction = 0x804003c4, kernel killed it.
[kernel] IllegalInstruction in application, kernel killed it.
[kernel] IllegalInstruction in application, kernel killed it.
[kernel] Panicked at src/trap/mod.rs:72: Unsupported trap Exception(LoadFault), stval = 0x18!
make[1]: Leaving directory '/home/w/Desktop/rcore/2023a-rcore-csline/os'
```

1. `bad_address`: 0地址访问
2. `bad_instruction`: 用户态使用 `sret` 指令(s模式返回u模式)导致异常
3. `bad_register`: 用户态访问 `sstatus` 寄存器导致异常

## 2

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. `a0`是 `__switch` 函数传入的TaskContext.  
`__restore` 第一个使用情景是作为内核层返回用户层的桥梁,第二种使用情景是启动用户态程序
2. `sepc`,`sscratch`,`sstatus`  
`sepc`是`sret`返回用户态的地址,`sscratch`是用户态堆栈地址,`sstatus`控制用户态的中断启用关闭,中断向量表地址,特权级SPP
3. `x2`是`sp`,下面有恢复`sp`的代码,`x4`是线程局部存储指针的寄存器,暂时还未用到
4. 该指令操作过程:

`crrw sp, sscratch, sp`



操作之后的`sp`是用户态的栈指针,`sscratch`是内核态的栈指针

5. `sret`,因为这是中断返回指令,`sstatus`的SPP位被设为0了
6. 操作之后的`sp`是内核态的栈指针,`sscratch`是用户态的栈指针
7. `ecall`

## 荣誉准则

---

交流对象: 群内大佬帮我解决了环境问题