第十一讲：处理机调度

第 7 节：rCore 调度框架

向勇、陈渝

清华大学计算机系

xyong,yuchen@tsinghua.edu.cn

2020 年 5 月 5 日

提纲

rCore 调度框架

rcore-thread/src/scheduler/mod.rs

```rust
/// The scheduler for a ThreadPool
pub trait Scheduler: 'static {
    /// Push a thread to the back of ready queue.
    fn push(&self, tid: Tid);
    /// Select a thread to run, pop it from the queue.
    fn pop(&self, cpu_id: usize) -> Option<Tid>;
    /// Got a tick from CPU.
    /// Return true if need reschedule.
    fn tick(&self, current_tid: Tid) -> bool;
    /// Set priority of a thread.
    fn set_priority(&self, tid: Tid, priority: u8);
    /// remove a thread in ready queue.
    fn remove(&self, tid: Tid);
}
```

与调度相关的线程控制函数

rcore-thread/src/thread_pool.rs

```rust
pub fn add(&self, mut context: Box<dyn Context>) -> Tid
pub(crate) fn tick(&self, cpu_id: usize, tid: Option<Tid>) -> bool
pub fn set_priority(&self, tid: Tid, priority: u8)
pub(crate) fn run(&self, cpu_id: usize) -> Option<(Tid, Box<dyn Context>)>
pub(crate) fn stop(&self, tid: Tid, context: Box<dyn Context>)
fn set_status(&self, tid: Tid, status: Status)
pub fn wakeup(&self, tid: Tid)
```

调度数据结构：struct ThreadPool

rcore-thread/src/thread_pool.rs

```
pub struct ThreadPool {
    threads: Vec<Mutex<Option<Thread>>>,
    scheduler: Box<dyn Scheduler>,
    timer: Mutex<Timer<Event>>,
}
```

调度算法和参数设置

rCore/kernel/src/process/mod.rs

```
pub fn init() {
    // NOTE: max_time_slice <= 5 to ensure 'priority' test pass
    let scheduler = scheduler::RRScheduler::new(5);
    let manager = Arc::new(ThreadPool::new(scheduler, MAX_PROCESS_NUM));
    unsafe {
        for cpu_id in 0..MAX_CPU_NUM {
            PROCESSORS[cpu_id].init(cpu_id, Thread::new_init(), manager.clone());
        }
    }
    crate::shell::add_user_shell();
    info!("process: init end");
}
```

时间片轮转（Round Robin）调度算法：数据结构

rcore-thread/src/scheduler/rr.rs

```rust
pub struct RRScheduler {
    inner: Mutex<RRSchedulerInner>,
}
struct RRSchedulerInner {
    max_time_slice: usize,
    infos: Vec<RRProcInfo>,
}
#[derive(Debug, Default, Copy, Clone)]
struct RRProcInfo {
    present: bool,
    rest_slice: usize,
    prev: Tid,
    next: Tid,
}
```

RR 调度算法实现

```rust
impl RRSchedulerInner {
    fn push(&mut self, tid: Tid) {…
    }

    fn pop(&mut self) -> Option<Tid> {…
    }

    fn tick(&mut self, current: Tid) -> bool {…
    }

    fn remove(&mut self, tid: Tid) {…
    }
}

impl RRSchedulerInner {
    fn _list_add_before(&mut self, i: Tid, at: Tid) {…
    }
    fn _list_add_after(&mut self, i: Tid, at: Tid) {…
    }
    fn _list_remove(&mut self, i: Tid) {…
    }
}
```

时间片用完时的线程调度和切换过程

```
/Users/xyong/github/rCore/kernel/src/arch/riscv/interrupt.rs
  pub extern "C" fn rust_trap(tf: &mut TrapFrame)
/Users/xyong/github/rCore/kernel/src/trap.rs
  pub fn timer()
/Users/xyong/github/rcore-thread/src/processor.rs
  pub fn tick(&self)
/Users/xyong/github/rcore-thread/src/processor.rs
  pub(crate) fn yield_now(&self)
```