实时 Linux 相关研究 (以 Preempt-rt 为例)

周元斌 @ 1 教 505

May 24, 2015

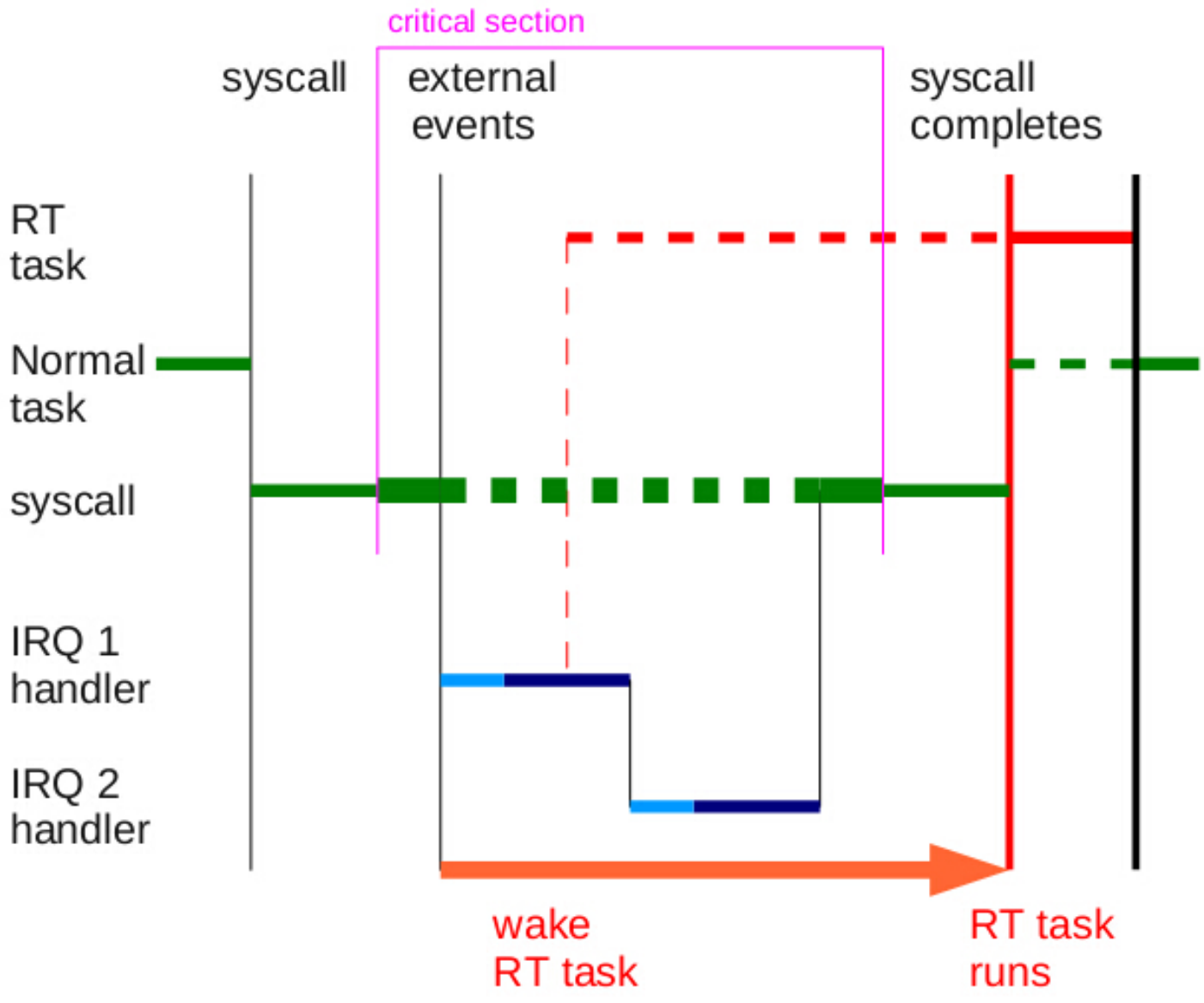
Introduction

什么是实时系统

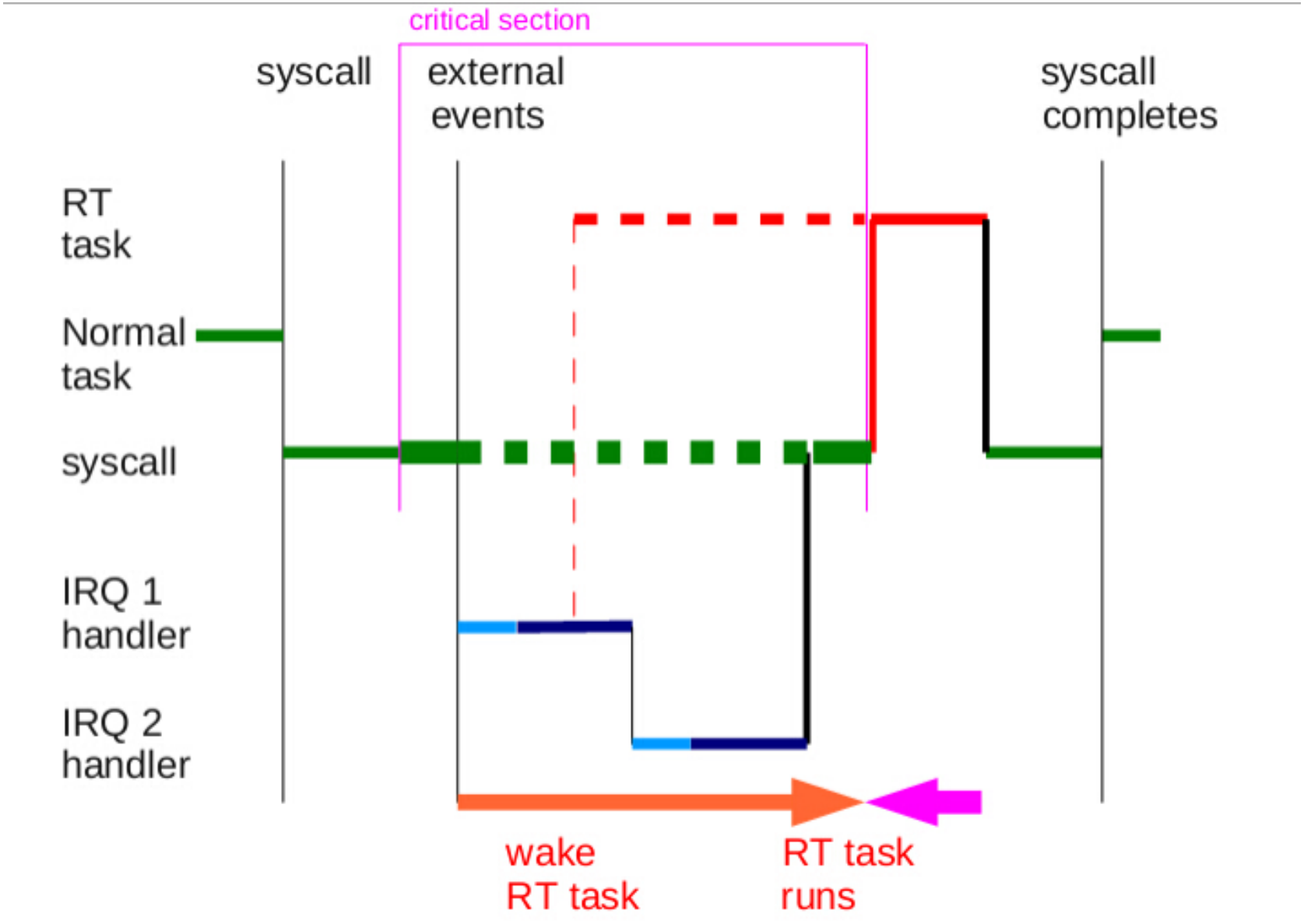
- 计算的正确性
  不仅仅依赖于逻辑的正确性，且依赖于产生结果的时间
- 如果系统的时间限制没有满足，系统就会发生故障

Application(Airbag of Car)

Non-preemptible Kernel



Preemptible Kernel



中断线程化

Linux 中断处理过程

- 分为顶半部分和底半部分
- 顶半部分负责保存现场和保存硬件数据（关中断）
- 底半部分对数据进行进一步的处理（开中断）
- 中断不一定是紧急事件
- 即使软硬件支持中断嵌套和中断优先级，但是中断能打断运行的任务，实时性的任务可能长期得不到运行

Preempt rt 中断线程化

- 通过中断线程化解决了中断优先级反转问题
- 主要思想：中断产生，唤醒中断化线程
- 实时任务的优先级高于中断化的线程，保证了实时任务的优先运行
- 对于时钟中断或者只需对数据进行简单处理的中断，不需要进行线程化处理（调度开销）

临界区可抢占

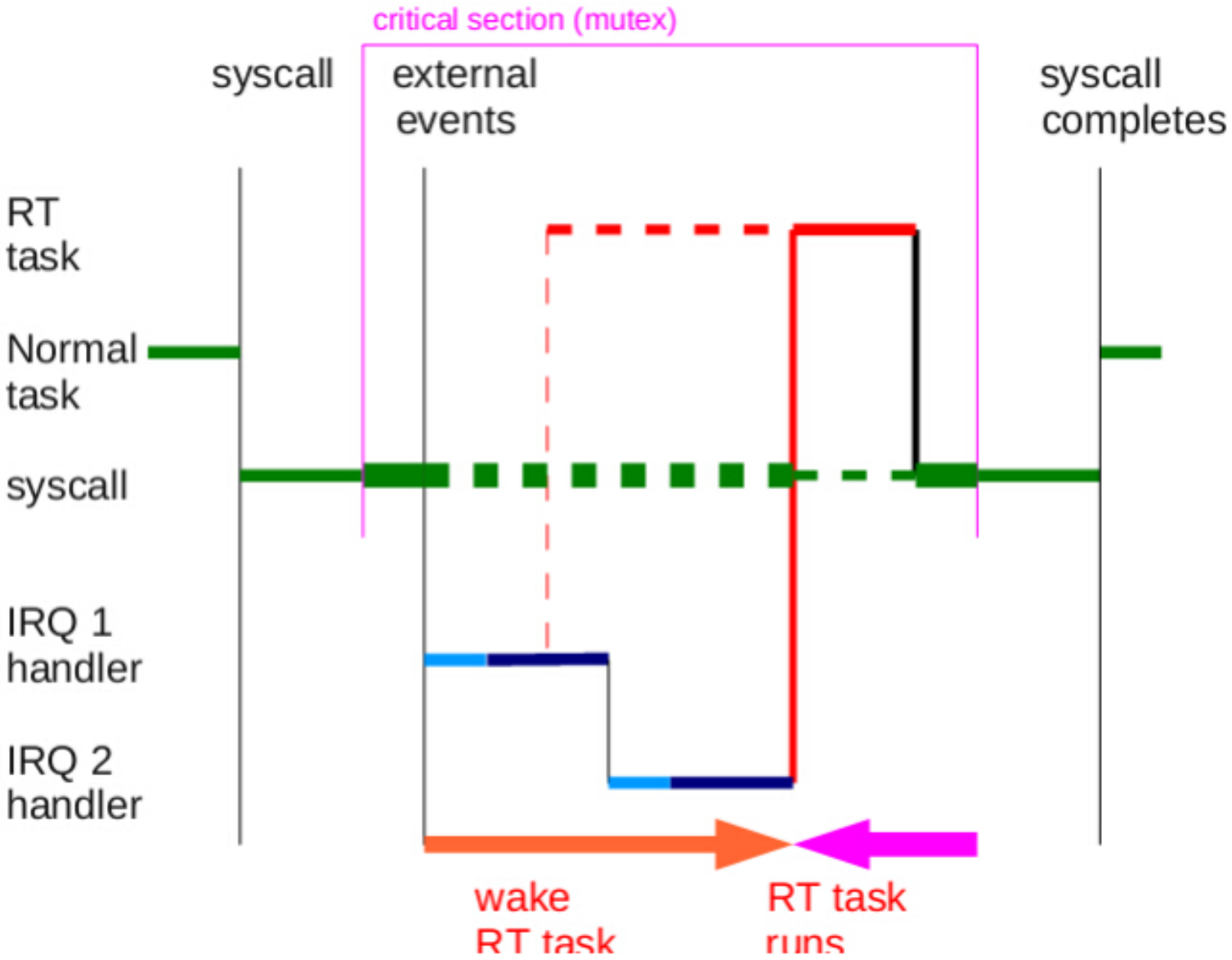
标准 Linux

- 内核中大量使用 spinlock(busy-waiting 类型的锁)
- 如果一个线程试图获得一个已被争用的 spinlock，该线程会一直自旋
- 由于 CPU 被强制等待，若加锁的线程执行时间很长，则会影响到系统的实时性

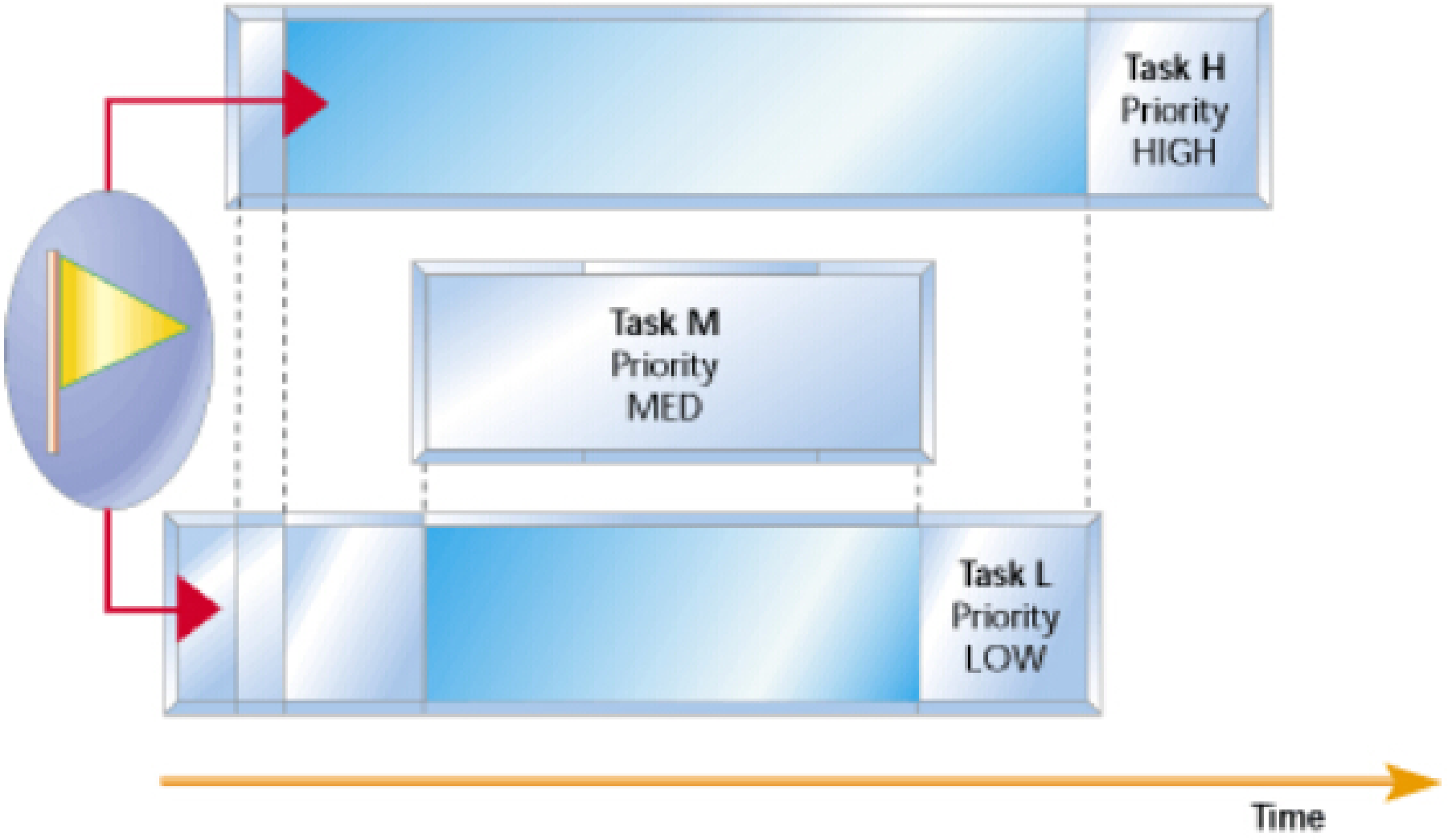
Preempt rt

- 使用 mutex（互斥锁）代替 spinlock（自旋锁）
- 一个线程要获取一个已经被占用的互斥锁，会把该线程挂起，进行上下文切换，到另外一个可以运行的线程
- 实时系统牺牲吞吐量来换取实时性

临界区可抢占


critical section (mutex)
syscall
external events
syscall completes
RT task
Normal task
syscall
IRQ 1 handler
IRQ 2 handler
wake RT task
RT task runs

优先级继承协议


Task H
Priority
HIGH
Task M
Priority
MED
Task L
Priority
LOW
Time

优先级继承协议

- 持有共享资源的线程将继承高优先级竞争者的优先级，使得能够优于中优先级线程运行
- 但释放资源后，该线程的优先级恢复到初始状态

高精度定时器

传统 Linux 做法

- 软时钟
- 通过周期性的滴答进行计时
- 当设置频率为 1kHz 时，时钟精度仅为 1ms
- 使用硬件时钟，硬件时钟频率可达百 MHz，精度可达 ns 级别
- 带来的问题是频繁的时钟中断

高精度定时器（已进入 mainline）

- 分为两个子系统，一个是 clock source，一个是 clock event
- clock source 用于获取系统时间
- clock event 可设置定时事件，任务可根据需要设置下一次时钟中断发生的时间（精度为 ns 级别）

Q&A