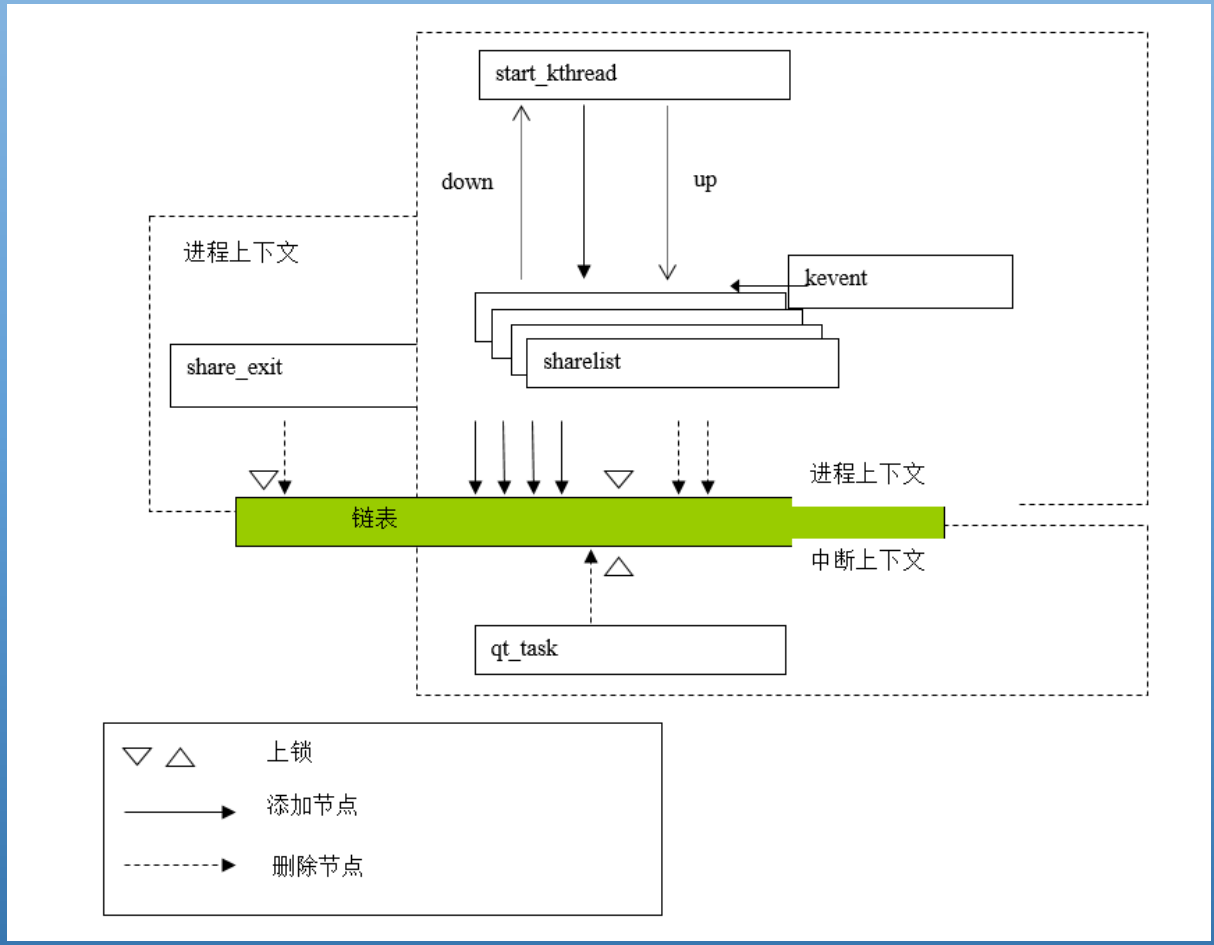
7.3动手实践-内核多任务并发实例

内核多任务并发——示意图


start_kthread
down
up
进程上下文
kevent
sharelist
share_exit
进程上下文
链表
中断上下文
qt_task
上锁
添加节点
删除节点

工作队列——基本概念

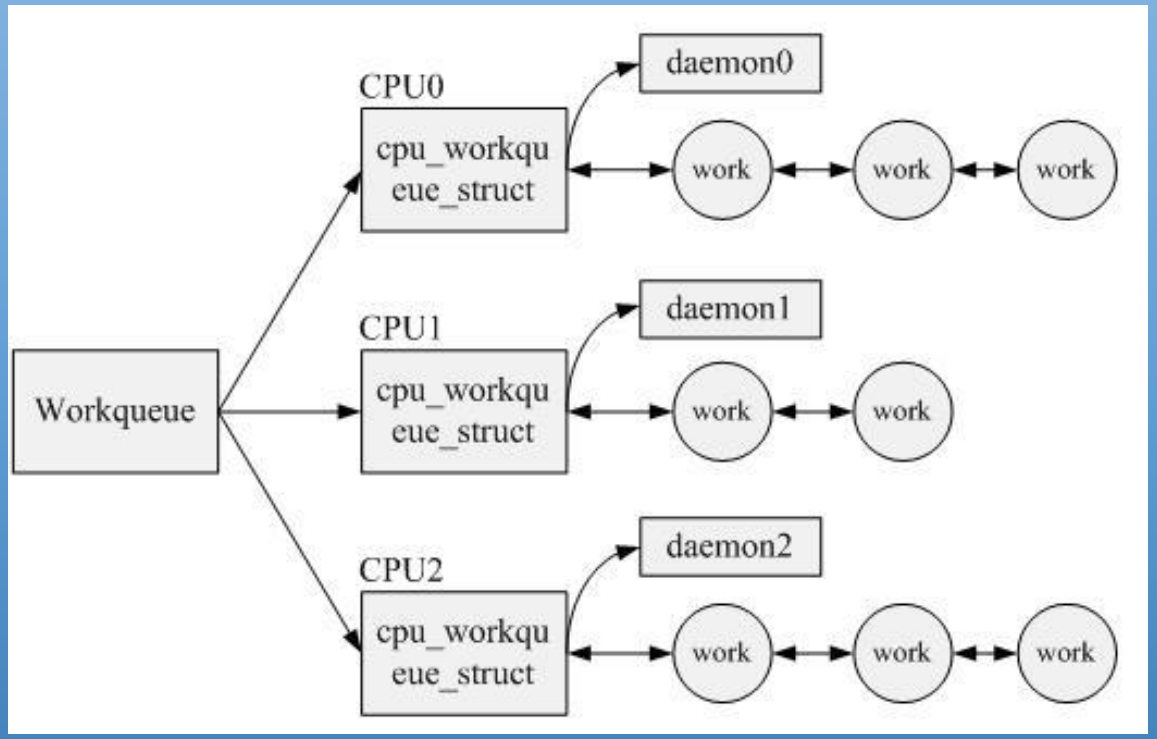
在内核代码中，经常希望延缓部分工作到将来某个时间执行。内核中提供了许多机制来实现这种延迟执行，例如可延迟函数，工作队列等；

工作队列（workqueue）可以把工作推后，并交由一个特殊的内核线程——工作者线程来执行，该线程运行在进程上文环境中，可以被阻塞。工作队列的优势在于它允许重新调度甚至是睡眠。

我们把推后执行的任务叫做工作（work），描述它的数据结构为work_struct

这些工作以队列结构组织成工作队列（workqueue），其数据结构workqueue_struct

工作队列——实现原理



工作队列内核实现

工作队列——常用API

- DECLARE_WORK（_work, _func）或INIT_WORK（_work, _func）
  用于初始化工作，指定一个工作_work所要执行性的函数_func；
- create_workqueue（name）
  用于创建一个workqueue队列，为系统中的每个CPU都创建一个内核线程。
  输入参数：@name：workqueue的名称
- create_singlethread_workqueue（name）
  用于创建workqueue，只创建一个内核线程。
  输入参数：@name：workqueue名称
- schedule_work
  调度执行一个具体的任务，执行的任务将会被挂入Linux系统提供的workqueue——keventd_wq
  输入参数：@work_struct：具体任务对象指针
- queue_work
  调度执行一个指定workqueue中的任务。
  输入参数：@workqueue_struct：指定的workqueue指针 @work_struct：具体任务对象指针
- destroy_workqueue
  释放workqueue队列。
  输入参数：@workqueue_struct：需要释放的workqueue队列指针

内核定时器

内核定时器，也称为动态定时器，是管理内核时间的基础，它是一种用来推迟执行程序的工具。

```
struct timer_list {
    /*
     * All fields that change during normal runtime grouped to
the
     * same cacheline
     */
    struct hlist_node    entry;
    unsigned long        expires;
    void                 (*function)(struct timer_list *);
    u32          flags;

#ifdef CONFIG_LOCKDEP
    struct lockdep_map  lockdep_map;
#endif
};
```

内核定时器

在Linux系统中，时钟分为硬件时钟（又叫实时时钟）和软件时钟（又叫系统时钟）。在对内核编程中，我们经常用到的是系统时钟。系统时钟的初始值在系统启动时，通过读取硬件时钟获得，然后由Linux内核来维护。

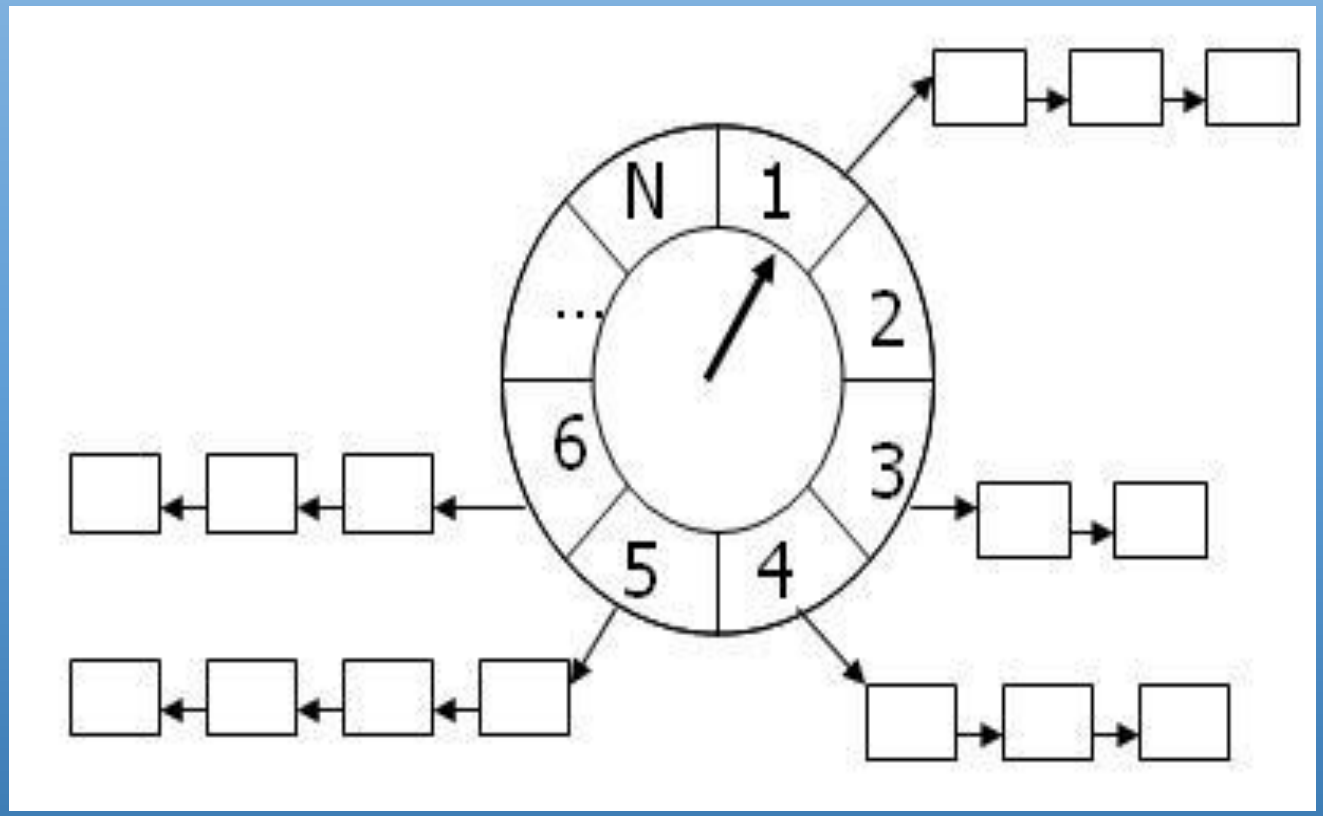
系统时钟以某种频率触发时钟中断，这个频率就称为节拍率（tick rate）。节拍率是通过静态预处理定义的，被定义为HZ，内核会根据HZ值，设置时钟事件，启动tick(节拍)中断。HZ表示1秒内产生多少个时钟硬件中断，tick就表示连续两个中断的间隔时间。

使用如下的命令，可以查看HZ数，HZ=250，所以在我的电脑上一个tick为4ms。

```
hawl29@ssssx:~$ cat /boot/config-`uname -r` | grep 'CONFIG_HZ='
CONFIG_HZ=250
```

而名为 jiffies 的全局变量，是用来记录系统启动以来经过的tick数。

内核定时器



基于时间轮算法的内核定时器

内核定时器——常用API

- DEFINE_TIMER(_name, _function)或timer_setup(timer, callback, flags)

  这两个宏是用来动态初始化一个内核定时器， 为内核timer的各个域赋值；
- add_timer(struct timer_list *timer)

  启动一个计时器,或者说激活一个定时器。

  @timer:要添加的定时器
- mod_timer(struct timer_list *timer, unsigned long expires)

  修改一个timer(定时器)的超时时间

  @timer:要修改的计时器 @expires:新的超时,单位jiffies
- del_timer(struct timer_list * timer)

  删除(停用)计时器。
  @timer:被停用的计时器

谢谢