2019 TencentOS tiny 物联网操作系统

学习永无止境~

——杰杰

关于我

一个走在物联网路上的小菜鸟~

博客：https://jiejietop.cn

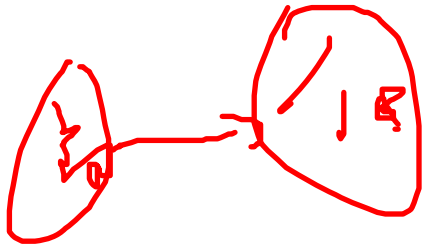
CSDN：https://blog.csdn.net/jiejiemcu

GitHub：https://github.com/jiejieTop

个人公众号

学习交流群：783234154

目录



01. 任务的状态

就绪态（K_TASK_STATE_READY）：该任务在就绪列表中，就绪的任务已经具备执行的能力，只等待调度器进行调度，新创建的任务会初始化为就绪态。

运行态（K_TASK_STATE_READY）：该状态表明任务正在执行，此时它占用处理器，其实此时的任务还是处于就绪列表中的，TencentOS调度器选择运行的永远是处于最高优先级的就绪态任务，当任务被运行的一刻，它的任务状态就变成了运行态。

睡眠态（K_TASK_STATE_SLEEP）：如果任务当前正在休眠让出CPU使用权，那么就可以说这个任务处于休眠状态，该任务不在就绪列表中，此时任务处于睡眠列表中（或者叫延时列表）。

等待态（K_TASK_STATE_PEND）：任务正在等待信号量、队列或者等待事件等状态。

挂起态（K_TASK_STATE_SUSPENDED）：任务被挂起，此时任务对调度器而言是不可见的。

退出态（K_TASK_STATE_DELETED）：该任务运行结束，并且被删除。

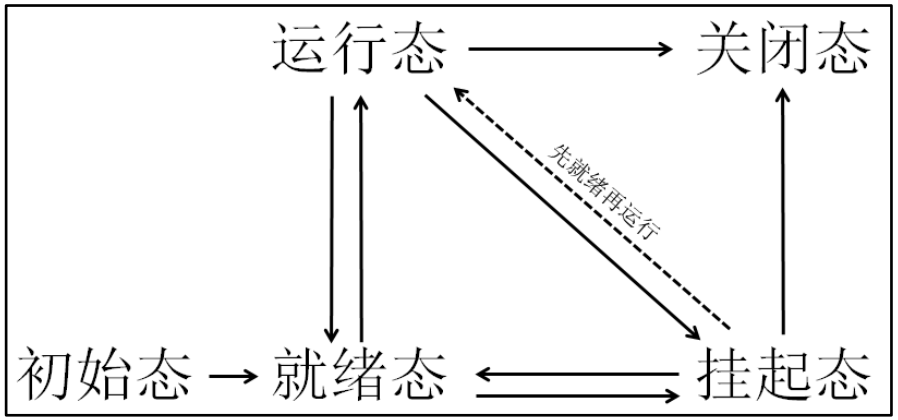
等待超时状态（K_TASK_STATE_PENDTIMEOUT）：任务正在等待信号量、队列或者等待事件发生超时的状态。

睡眠挂起态（K_TASK_STATE_SLEEP_SUSPENDED）：任务在睡眠中被挂起时的状态。

等待挂起态（K_TASK_STATE_PEND_SUSPENDED）：任务正在等待信号量、队列或者等待事件时被挂起的状态。

等待超时挂起态（K_TASK_STATE_PENDTIMEOUT_SUSPENDED）：任务正在等待信号量、队列或者等待事件发生超时，但此时任务已经被挂起的状态。

02. 任务状态迁移



03. TencentOS tiny中维护任务的数据结构

就绪列表 k_rdyq：用于挂载系统中的所有处于就绪态的任务

延时列表 k_tick_list：与系统时间相关的任务都会被挂载到这个列表中，可能是睡眠、有期限地等待信号量、事件、消息队列等情况~

04. 任务控制块

在多任务系统中，任务的执行是由系统调度的。系统为了顺利的调度任务，为每个任务都额外定义了一个任务控制块，这个任务控制块就相当于任务的身份证，里面存有任务的所有信息，比如任务的栈指针，任务名称，任务的形参等。有了这个任务控制块之后，以后系统对任务的全部操作都可以通过这个任务控制块来实现。

05. 任务创建 tos_task_create()

在TencentOS tiny中，凡是使用__API__修饰的函数都是提供给用户使用的，而使用__KERNEL__修饰的代码则是给内核使用的。

```
__API__ k_err_t tos_task_create(k_task_t *task, char *name, k_task_entry_t entry, void *arg, k_prio_t prio,
                                k_stack_t *stk_base, size_t stk_size, k_timeslice_t timeslice)
```

TencentOS的创建任务函数有好几个参数：

参数	含义
task	任务控制块
name	任务名字
entry	任务主体
arg	任务形参
prio	优先级
stk_base	任务栈基地址
stk_size	任务栈大小
timeslice	时间片

06.任务销毁 tos_task_destroy()

这个函数十分简单，根据传递进来的任务控制块销毁任务，也可以传递进NULL表示销毁当前运行的任务。但是不允许销毁空闲任务k_idle_task，当调度器被锁住时不能销毁自身，会返回K_ERR_SCHED_LOCKED错误代码。如果使用了互斥量，当任务被销毁时会释放掉互斥量，并且根据任务所处的状态进行销毁，比如任务处于就绪态、延时态、等待态，则会从对应的状态列表中移除。

07.任务睡眠 tos_task_delay ()

任务睡眠非常简单，主要的思路就是将任务从就绪列表移除，然后添加到延时列表中k_tick_list，如果调度器被锁，直接返回错误代码K_ERR_SCHED_LOCKED，如果睡眠时间为0，则调用tos_task_yield函数发起一次任务调度；调用tick_list_add函数将任务插入延时列表中，睡眠的时间delay是由用户指定的。不过需要注意的是如果任务睡眠的时间是永久睡眠TOS_TIME_FOREVER，将返回错误代码K_ERR_DELAY_FOREVER，这是因为任务睡眠是主动行为，如果永久睡眠了，将没法主动唤醒，而任务等待事件、信号量、消息队列等行为是被动行为，可以是永久等待，一旦事件发生了、信号量呗释放、消息队列不为空时任务就会被唤醒，这是被动行为，这两点需要区分开来。最后调用readyqueue_remove函数将任务从就绪列表中移除，然后调用knl_sched函数发起一次任务调度，就能切换另一个任务。

06.任务销毁 tos_task_destroy()

这个函数十分简单，根据传递进来的任务控制块销毁任务，也可以传递进NULL表示销毁当前运行的任务。但是不允许销毁空闲任务k_idle_task，当调度器被锁住时不能销毁自身，会返回K_ERR_SCHED_LOCKED错误代码。如果使用了互斥量，当任务被销毁时会释放掉互斥量，并且根据任务所处的状态进行销毁，比如任务处于就绪态、延时态、等待态，则会从对应的状态列表中移除。

07.任务挂起、恢复

挂起指定任务。被挂起的任务绝不会得到CPU的使用权，不管该任务具有什么优先级。

任务可以通过调用tos_task_suspend()函数都可以将处于任何状态的任务挂起，被挂起的任务得不到CPU的使用权，也不会参与调度，它相对于调度器而言是不可见的，除非它从挂起态中恢复。

任务恢复就是让挂起的任务重新进入就绪状态，恢复的任务会保留挂起前的状态信息，在恢复的时候根据挂起时的状态继续运行。如果被恢复任务在所有就绪态任务中，处于最高优先级列表的第一位，那么系统将进行任务上下文的切换。

任务恢复的函数是：tos_task_resume()

08. 任务设计要点

中断上下文环境：不能使用挂起当前任务的操作，不允许调用任何会阻塞运行的API函数接口。

任务上下文环境：任务中不允许出现死循环（此处的死循环是指没有不带阻塞机制的任务循环体），将紧急的处理事件的任务优先级设置得高一些。

空闲任务：不允许创建与空闲任务相同优先级的任务

任务的执行时间：程序运行时间、周期。

06. 写代码做实验

创建两个任务

THANKS