FreeRTOS学习笔记（四）

FreeRTOS中的列表和列表项

- 列表：列表是FreeRTOS中的一个数据结构，概念上和链表有点类似，列表被用来跟踪FreeRTOS中的任务。
- 列表项：存放在列表中的项目。
- 迷你列表项：相对于列表项少了几个成员变量，满足特殊情况下面的需求，节省内存。

FreeRTOS中列表和列表项的API函数

```
/**
* 描述：初始化函数是用来初始化列表结构体List_t中的成员变量
*       初始化之后只有一个列表项，那就是xListEnd
* 参数：一个List_t类型的列表
*/
void vListInitialise(List_t * const pxList);

/**
* 描述：初始化列表项函数
* 参数：一个Listitem_t类型的列表项
*/
void vListInitialiseItem(ListItem_t * const pxItem);

/**
* 描述：插入函数（具体是插入什么位置，是由xItemValue来决定的）
* 参数：pxList要插入到的列表
*       pxNewListItem要插入的列表项
*/
void vListInsert(List_t * const pxList, ListItem_t * const pxNewListItem);

/**
* 描述：插入到尾部函数
* 参数：pxList要插入到的列表
*       pxNewListItem要插入的列表项
*/
void vListInsertEnd(List_t * const pxList, ListItem_t * const pxNewListItem);

/**
* 描述：删除列表项
* 参数：pxItemToRemove需要删除的列表项
* 返回值：列表剩余的列表项数目
*/
UBaseType_t uxListRemove( ListItem_t * const pxItemToRemove );
```

FreeRTOS任务创建和调度器的开启

- 创建任务的过程
    1. 给任务堆栈申请内存
    2. 如果上一步成功，那就给任务控制块申请内存
    3. 上一步成功之后，初始化内存控制块中的任务堆栈字段
    4. 如果任务控制块申请不成功，就释放第一步申请到的任务堆栈内存
    5. 标记任务堆栈和任务控制块是使用动态内存分配方法得到的
    6. 使用prvInitialiseNewTask（ ）初始化任务，完成任务控制块中各个字段的初始化工作
    7. 使用prvAddNewTaskToReadyList（ ）将任务加入到就绪表中

- 调度器开启的过程（vTaskStartScheduler（ ）；）
    1. 创建空闲任务（空闲任务的优先级最低）
    2. 若要使用软件定时器的话，需要通过xTimerCreateTimerTask（ ）来创建定时器服务任务
    3. 关闭中断
    4. 变量xSchedulerRunning设置为pdTRUE，表示调度器开始运行
    5. configGENERATE_RUN_TIME_STATS被设置为1（使能了时间统计功能），portCONFIGURE_TIMER_FOR_RUN_TIME_STATS（ ）；需要用户自行实现，配置一个定时器
    6. 使用xPortStartScheduler（ ）；来初始化跟调度器启动有关的硬件（滴答定时器，FPU单元，PendSV中断等等）

- 空闲任务
    1. 满足任务调度器启动以后至少有一个任务运行
    2. 判断是否有任务删除，若有，释放被删除任务的任务堆栈和任务控制块的内存
    3. 运行用户设置的空闲任务钩子函数
    4. 判断是否开启低功耗tickless模式，如果开启，还需要做相应的处理

FreeRTOS**任务切换**

- 通过向中断控制和状态寄存器ICSR的bit28写入1挂起PendSV来启动PendSV中断
- 可以使用taskYIELD（ ）来启动系统调用（使其进入PendSV中断）
- 定时器中断也是通过触发PendSV中断来完成任务调度的
- 调度过程
    1. 保存现场以及相关的变量
    2. 查找下一个要运行的任务（硬件方法，软件方法）

- 时间片轮转调度
    1. 使用滴答定时器来触发任务调度
    2. 通过设置宏configTICK_RATE_HZ来设置滴答定时器的中断周期
    3. configUSE_PREEMPTION 和 configUSE_TIME_SLICING 两个宏都设置为1，才可以使用时间片轮转调度

```
void xPortSysTickHandler( void )
{
	vPortRaiseBASEPRI();
	{
		/* 如果当前任务所对应的优先级下还有其他任务，返回pdTRUE */
		if( xTaskIncrementTick() != pdFALSE )
		{
	/* 触发PendSV中断 */
			portNVIC_INT_CTRL_REG = portNVIC_PENDSVSET_BIT;
		}
```

```
        }
        vPortClearBASEPRIFromISR();
}
```