# 阿熊的FreeRTOS教程系列！

哈喽大家好！我是你们的老朋友阿熊！STM32教程系列更新完结已经有一段时间了，视频反馈还是不错的，从今天开始我们将会更新我们的FreeRTOS的教程

由于东西真的太多了，也纠结了很久要不要讲这个系列，毕竟难度真的很大，怕在难以做到那么通俗易懂，经过一段时间的考虑，还是决定好了给大家做一个入门级的讲解使用，由于FreeRTOS的内容真的很多，作为还是学生的我使用的也相对较少，操作系统层面的东西，我会用最大的能力去让大家理解，主要讲述主要功能，学完以后保证大伙可以理解80%以上的FreeRTOS的使用场景，好了废话不多说，开始我们的课程吧！


## 第三章：**FreeRTOS**优先级讲解

我们的日常生活中干什么事情也会有一个轻重缓急，而我们的操作系统同样的它也会有一个轻重缓急，也就是优先级

也就是我们前面创建任务给定的优先级

```
xTaskCreate(vTask1,"LED1",128,NULL,1,NULL);
```

其中第五个参数，就是我们的优先级，默认情况下其范围是0~(configMAX_PRIORITIES – 1)（configMAX_PRIORITIES =32），绝大多数情况下，他都是够用的，毕竟我们的操作系统是支持同优先级的

这里提一嘴，理论上我们的**FreeRTOS**，如果仅通过软件层面上是可以实现无穷多个优先级的，但是这里结合了我们架构相关的东西，默认他最大只支持**32**级，这里大概了解一下就可以了，反正正常情况下我们是用不到那么多优先级的

这里还需要和小伙伴们讲一下我们的FreeRTOS的任务执行原则

## 壹：**FreeRTOS**执行任务的原则：

默认情况为：使用时间片抢占式任务调度（合作式调度模式）

合作式任务调度解析：

仅作了解即可，使用此种调度方式时，我们的FreeRTOS不会主动去进行任务的切换，而是当任务进行阻塞状态时，才会有那个任务的切换

合作式调度模式几乎不会用到，系统也没有相应的更新维护了

抢占式任务调度解析：

各个任务执行的时候，就像我们裸机开发中的中断一样，是需要去抢这个名额，才能有机会执行，永远先执行优先级最高的任务，优先级最高的任务执行完了才有机会让低等级的任务去执行

第一原则：永远先执行优先级最高的任务

抢占式调度任务又可以分为以下两种：

使用时间片的抢占式调度方法：

同优先级的情况下，我们每一个时间片他都会去判断一下要不要切换任务或者说会主动去切换任务

不使用时间片的抢占式调度方法：

同优先级的情况下，将不会去判断一下要不要切换任务或者说不会主动切换任务

## 贰：**FreeRTOS**优先级示例分析：

这里的示例都是默认配置下去验证的（使用时间片抢占式任务调度）并且使用的是上一章相同例程

### ①：同优先级两个任务

现象：两个任务都正常运行，不会有干扰

分析：使用时间片抢占式任务调度的模式，同优先级情况下，每个时间片都会进行任务切换

### ②：**LED1**优先级高于**LED2**

现象：只有**LED1**任务在执行，**LED2**任务无法正常执行

分析：**LED1**任务的优先级高于**LED2**任务的优先级，我们系统运行的第一原则：永远先执行优先级最高的任务所以就把我们的**LED2**任务给饿死了，每个时间片都无法抢占到执行任务的机会，所以就永远无法执行

### ③：再②的基础上将**Task1**中的**HAL_Delay()**换为**vTskDelay()[osDelay()]**

现象：任务正常运行，现象和①一样

分析：**vTskDelay()[osDelay()]**有着某种能力使得我们的高优先级的任务停下来，然后我们低优先级的任务有执行的机会

剧透一下，这种状态，叫做“阻塞”，顾名思义使用以后会进入一种堵车的状态，就比如说我们的vTskDelay(1000);就是堵车了一秒，这一秒钟的任务执行权力就被放开了，这样的话我们的低优先级的任务才有机会进行

好了，本期视频的内容就是这些，我们下一章节就会开始讲一下我们任务的各种状态