

# OpenHarmony内核开发 任务管理



# 目录

## CONTENTS

- [ 01 ] 什么是任务
- [ 02 ] 任务调度机制
- [ 03 ] 任务的相应接口
- [ 04 ] 如何创建和删除任务

# 01

## 什么是任务

- 从OpenHarmony内核的角度看，**任务是竞争系统资源的最小运行单元**。任务可以使用或等待CPU、使用内存空间等系统资源，并独立于其它任务运行。
- 在LiteOS中，任务模块可以给用户提供多个任务，实现了任务之间的切换和通信，帮助用户管理业务程序流程。这样用户可以将更多的精力投入到业务功能的实现中。
- **LiteOS中的任务是抢占式调度机制，高优先级的任务可打断低优先级任务**，低优先级任务必须在高优先级任务阻塞或结束后才能得到调度，同时支持时间片轮转调度方式。
- LiteOS的**任务默认有32个优先级(0-31)，最高优先级为0，最低优先级为31**。

## 02

# 任务调度机制

一般而言，任务状态通常分为以下四种：

- (1) 就绪状 (Ready)：该任务在就绪列表中，只等待CPU调用。
- (2) 运行状 (Running)：该任务正在CPU中执行。
- (3) 阻塞状 (Blocked)：该任务不在就绪列表中。该任务被挂起、任务被延时、任务正在等待信号量、读写队列或者等待读写事件等。
- (4) 退出态 (Dead)：该任务运行结束，等待系统回收资源。

## 03

# 任务的接口

任务接口的头文件

/kernel/liteos\_m/kernel/include/los\_task.h

OpenHarmony内核开发中，任务接口有很多，主要分为几大类：

- (1) 创建和删除任务；
- (2) 控制任务状态；
- (3) 控制任务调度；
- (4) 控制任务优先级；
- (5) 获取任务信息。

## 03

## 任务的接口

功能分类	接口名	功能描述
创建和删除任务	LOS_TaskCreateOnly	创建任务，并使该任务进入suspend状态，不对该任务进行调度。
	LOS_TaskCreate	创建任务，并使该任务进入ready状态。如果就绪队列中没有更高优先级的任务，则运行该任务。
	LOS_TaskDelete	删除指定的任务
控制任务状态	LOS_TaskResume	恢复挂起的任务，使该任务进入ready状态。
	LOS_TaskSuspend	挂起指定的任务，然后切换任务
	LOS_TaskDelay	任务延时等待，释放CPU，等待时间到期后该任务会重新进入ready状态。传入参数为Tick数目。
	LOS_Msleep	传入参数为毫秒数，转换为Tick数目，调用LOS_TaskDelay。
	LOS_TaskYield	当前任务时间片设置为0，释放CPU，触发调度运行就绪任务队列中优先级最高的任务。
控制任务调度	LOS_TaskLock	锁任务调度，但任务仍可被中断打断。
	LOS_TaskUnlock	解锁任务调度。
	LOS_Schedule	触发任务调度。

## 03

# 任务的接口

功能分类	接口名	功能描述
控制任务优先级	LOS_CurTaskPriSet	设置当前任务的优先级。
	LOS_TaskPriSet	设置指定任务的优先级。
	LOS_TaskPriGet	获取指定任务的优先级。
获取任务信息	LOS_CurTaskIDGet	获取当前任务的ID。
	LOS_NextTaskIDGet	获取任务就绪队列中优先级最高的任务的ID。
	LOS_NewTaskIDGet	等同LOS_NextTaskIDGet。
	LOS_CurTaskNameGet	获取当前任务的名称
	LOS_TaskNameGet	获取指定任务的名称。
	LOS_TaskStatusGet	获取指定任务的状态。
	LOS_TaskInfoGet	获取指定任务的信息，包括任务状态、优先级、任务栈大小、栈顶指针SP、任务入口函数、已使用的任务栈大小等。
	LOS_TaskIsRunning	获取任务模块是否已经开始调度运行。
任务信息维测	LOS_TaskSwitchInfoGet	获取任务切换信息，需要开启宏LOSCFG_BASE_CORE_EXC_TSK_SWITCH。

## 03

## 任务的接口

`UINT32 LOS_TaskCreate(UINT32 *taskID, TSK_INIT_PARAM_S *taskInitParam);`

该函数主要功能是创建任务，并使该任务进入ready状态。如果就绪队列中没有更高优先级的任务，则运行该任务。其中，参数taskID表示任务的句柄，参数taskInitParam则是任务的一些详细信息。

```
/**
 * @ingroup los_task
 * Define the structure of the parameters used for task creation.
 *
 * Information of specified parameters passed in during task creation.
 */
typedef struct tagTaskInitParam {
    TSK_ENTRY_FUNC pfnTaskEntry;    /**< Task entrance function */
    UINT16 usTaskPrio;              /**< Task priority */
    UINT32 uwArg;                   /**< Task parameters */
    UINT32 uwStackSize;             /**< Task stack size */
    CHAR *pcName;                   /**< Task name */
    UINT32 uwResved;                /**< Reserved */
} TSK_INIT_PARAM_S;
```



## 03

## 任务的接口

```
typedef struct tagTskInitParam {  
    TSK_ENTRY_FUNC pfnTaskEntry;           /**< Task entrance function */  
    UINT16 usTaskPrio;                      /**< Task priority */  
    UINT32 uwArg;                          /**< Task parameters */  
    UINT32 uwStackSize;                   /**< Task stack size */  
    CHAR *pcName;                         /**< Task name */  
    UINT32 uwResved;                      /**< Reserved */  
} TSK_INIT_PARAM_S;
```

- ❑ pfnTaskEntry: 任务执行函数入口。
- ❑ usTaskPrio: 任务的优先级。一般为: 0~31。LiteOS的任务默认有32个优先级(0-31), 最高优先级为0, 最低优先级为31。
- ❑ uwArg: 任务执行函数的参数。
- ❑ uwStackSize: 任务堆栈的大小。根据任务自身的实际情况而定。
- ❑ pcName: 任务的名称。
- ❑ uwResved: 保留。

## 03

# 任务的接口

`UINT32 LOS_TaskDelete(UINT32 taskID);`

该函数主要功能是删除指定的任务。参数taskID为任务的句柄。

`VOID LOS_Msleep(UINT32 mSecs);`

该函数主要功能是任务进入睡眠。参数mSecs为毫秒数，转换为Tick数目，调用LOS\_TaskDelay。

## 04

# 如何创建和删除任务

### 1、打开sdk下面路径的文件

```
vendor/lockzhiner/rk2206/samples/a1_kernal_task/kernel_task_example.c
```

### 2、创建任务

在task\_example函数中，通过LOS\_TaskCreate函数创建task\_one和task\_two两个任务。

```
void task_example()
{
    unsigned int thread_id1;
    unsigned int thread_id2;
    TSK_INIT_PARAM_S task1 = {0};
    TSK_INIT_PARAM_S task2 = {0};
    unsigned int ret = LOS_OK;
```

## 如何创建和删除任务

```
task1.pfnTaskEntry = (TSK_ENTRY_FUNC)task_one;
```

```
task1.uwStackSize = 2048;
```

```
task1.pcName = "Task_One";
```

```
task1.usTaskPrio = 24;
```

```
ret = LOS_TaskCreate(&thread_id1, &task1);
```

```
if (ret != LOS_OK)
```

```
{
```

```
printf("Falied to create Task_One ret:0x%x\n", ret);
```

```
return;
```

```
}
```

```
task2.pfnTaskEntry = (TSK_ENTRY_FUNC)task_two;
```

```
task2.uwStackSize = 2048;
```

```
task2.pcName = "Task_Two";
```

```
task2.usTaskPrio = 25;
```

```
ret = LOS_TaskCreate(&thread_id2, &task2);
```

```
if (ret != LOS_OK)
```

```
{
```

```
printf("Falied to create Task_Two ret:0x%x\n", ret);
```

```
return;
```

```
}
```

```
}
```



## 如何创建和删除任务

task\_one函数每1秒执行一次打印日志。

```
void task_one()  
{  
    while (1)  
    {  
        printf("This is %s\n", __func__);  
        LOS_Msleep(1000);  
    }  
}
```

task\_two函数每2秒执行一次打印日志。

```
void task_two()  
{  
    while (1)  
    {  
        printf("This is %s\n", __func__);  
        LOS_Msleep(2000);  
    }  
}
```

# 如何创建和删除任务

## 3、修改编译脚本

修改 `vendor/lockzhiner/rk2206/sample` 路径下 `BUILD.gn` 文件，指定 `a1_kernal_task` 参与编译。

```
"/a1_kernal_task:task_example",
```

修改 `device/lockzhiner/rk2206/sdk_liteos` 路径下 `Makefile` 文件，添加 `-ltask_example` 参与编译。

```
hardware_LIBS = -lhal_iohardware -lhardware -ltask_example
```

## 4、编译固件

```
hb set -root .
```

```
hb set
```

```
hb build -f
```

# 如何创建和删除任务

## 5、烧写固件

## 6、通过串口查看结果

运行结果

```
This is task_one
```

```
This is task_one
```

```
This is task_two
```

```
This is task_one
```

```
This is task_one
```

```
This is task_two
```

```
.....
```

# 谢谢聆听

单击此处添加副标题内容