HarmonyOS内核开发—事件管理

前言

本节主要介绍：

- 事件的相关概念
- 事件的运作机制
- 如何利用事件进行任务间同步

目录

事件基本概念

事件是一种实现任务间通信的机制，可用于实现任务间的同步，但事件通信只能是事件类型的通信，无数据传输。一个任务可以等待多个事件的发生：可以是任意一个事件发生时唤醒任务进行事件处理;也可以是几个事件都发生后才唤醒任务进行事件处理。事件集合用32位无符号整型变量来表示，每一位代表一个事件。

多任务环境下，任务之间往往需要同步操作。事件可以提供一对多、多对多的同步操作。一对多同步模型：一个任务等待多个事件的触发；多对多同步模型：多个任务等待多个事件的触发。

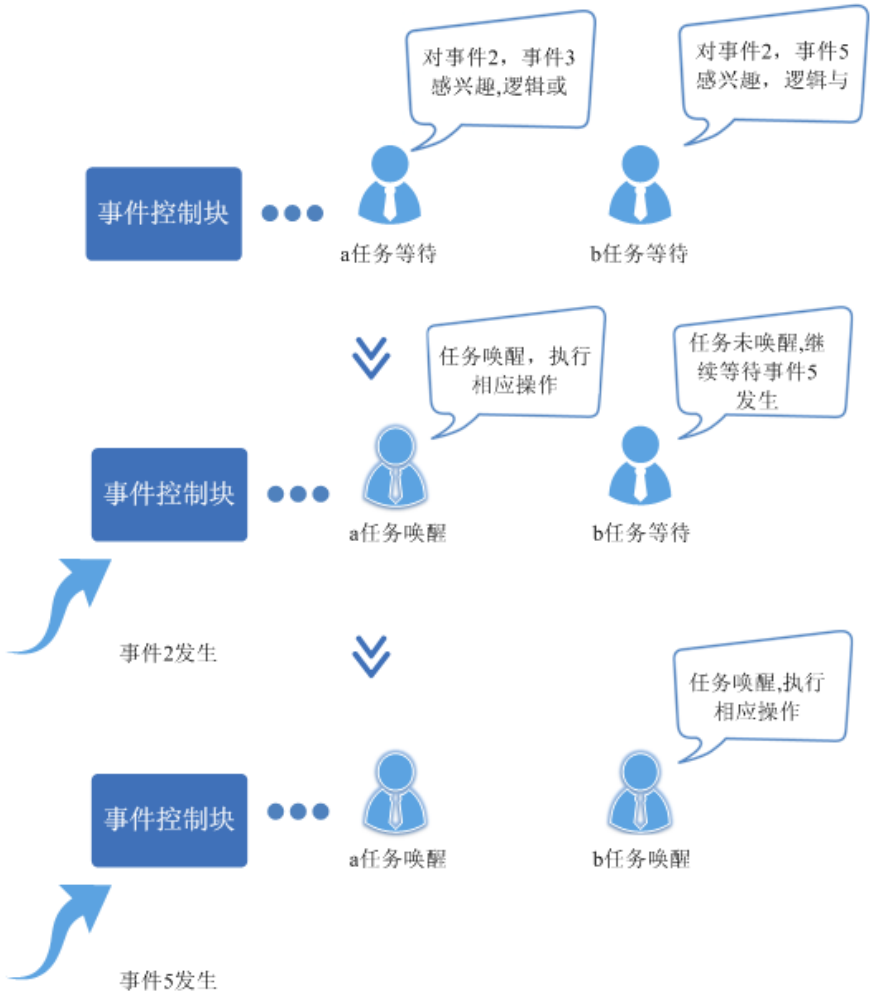
任务可以通过创建事件控制块来实现对事件的触发和等待操作。LiteOS的事件仅用于任务间的同步，

事件运作机制

读事件时，可以根据入参事件掩码类型uwEventMask读取事件的单个或者多个事件类型。事件读取成功后，如果设置LOS_WAITMODE_CLR会清除已读取到的事件类型，反之不会清除已读到的事件类型，需显式清除。可以通过入参选择读取模式，读取事件掩码类型中所有事件还是读取事件掩码类型中任意事件。

写事件时，对指定事件写入指定的事件类型，可以一次同时写多个事件类型。写事件会触发任务调度。

清除事件时，根据入参事件和待清除的事件类型，对事件对应位进行清0操作。



实现事件功能

cmsis_os2的API事件接口简介：

接口名	功能描述
osEventFlagsNew	创建事件标记对象
osEventFlagsSet	设置事件标记
osEventFlagsWait	等待事件标记触发
osEventFlagsDelete	删除事件标记对象

创建事件标记对象：osEventFlagsNew (const osEventFlagsAttr_t *attr);

设置事件标记：osEventFlagsSet (osEventFlagsId_t ef_id, uint32_t flags);

等待事件标记触发：osEventFlagsWait (osEventFlagsId_t ef_id, uint32_t flags, uint32_t options, uint32_t timeout);

删除事件标记对象：osEventFlagsDelete (osEventFlagsId_t ef_id);

软件定时器扩展实验

扩展实验代码

```
/***** 发送事件 *****/
void Thread_EventSender(void *argument)
{
  (void)argument;
  while (1)
  {
    osEventFlagsSet(evt_id, FLAGS_MSK1);
    osEventFlagsSet(evt_id, FLAGS_MSK2);
    osEventFlagsSet(evt_id, FLAGS_MSK3);

    //suspend thread
    osThreadYield();

    osDelay(100);
  }
}
```

```
/***** 接收事件 *****/
void Thread_EventReceiver(void *argument)
{
  (void)argument;
  uint32_t flags;

  while (1)
  {
    flags = osEventFlagsWait(evt_id, FLAGS_MSK1|FLAGS_MSK2|FLAGS_MSK3, osFlagsWaitAll, osWaitForever);
    printf("Receive Flags is %d\n", flags);
  }
}
```

本节小结

- 1、了解事件的概念
- 2、掌握如何创建和设置事件标记
- 3、掌握如何使用多个事件同步一个任务

谢谢观看

开源从小熊派开始

OPEN-SOURCE STARTED WITH THE BEARPI