



HarmonyOS内核开发—互斥锁

本节主要介绍:

- 互斥锁的相关概念
- 互斥锁的运作机制
- 如何利用互斥锁实现对共享资源的独占式处理

三 目录

1. 互斥锁基本概念
2. 互斥锁运作机制
3. 实现互斥锁功能
4. 互斥锁扩展实验
5. 总结



互斥锁基本概念

互斥锁的概念：

- 1、互斥锁又称**互斥型信号量**，是一种特殊的**二值性信号量**，用于实现对共享资源的**独占式**处理。
 - 2、任意时刻互斥锁的状态只有两种：**开锁或闭锁**。
 - 3、当有**任务持有时**，**互斥锁处于闭锁状态**，这个任务获得该互斥锁的所有权。
 - 4、当该**任务释放时**，**该互斥锁被开锁**，任务失去该互斥锁的所有权。
 - 5、当一个任务持有互斥锁时，其他任务将不能再对该互斥锁进行开锁或持有。
 - 6、多任务环境下往往存在多个任务竞争同一共享资源的应用场景，互斥锁可被用于对共享资源的保护从而实现独占式访问。
- 另外，互斥锁可以解决信号量存在的优先级翻转问题。

LiteOS提供的互斥锁具有如下特点：

通过优先级继承算法，解决优先级翻转问题。

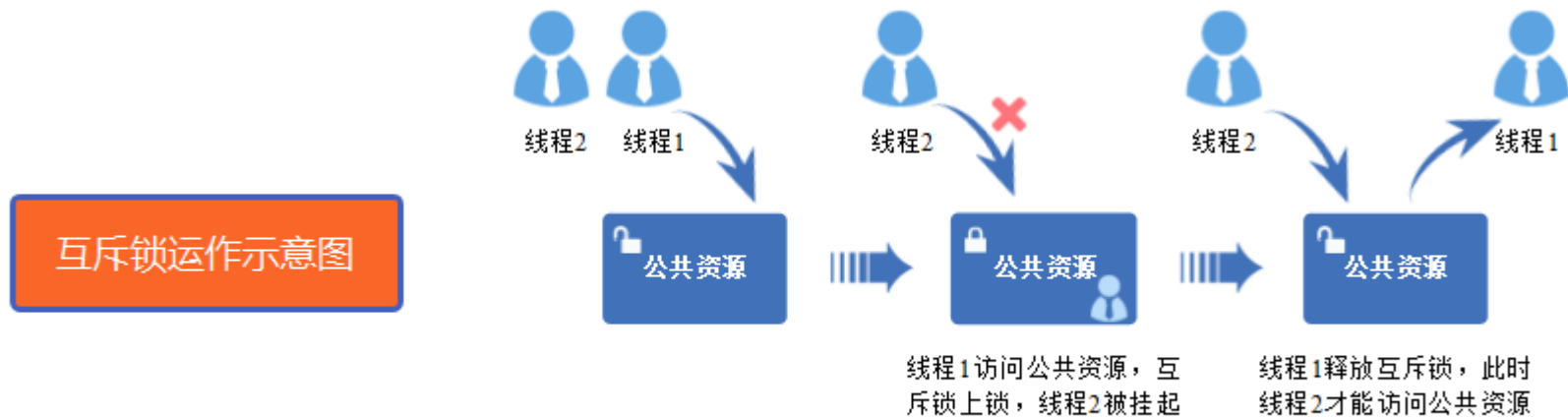


互斥锁运作机制

运作原理

多任务环境下会存在**多个任务访问同一公共资源**的场景，而有些公共资源是非共享的，**需要任务进行独占式处理**。互斥锁怎样来避免这种冲突呢？

用互斥锁处理非共享资源的同步访问时，**如果有任务访问该资源，则互斥锁为加锁状态**。此时其他任务如果想访问这个公共资源则会被阻塞，直到互斥锁被持有该锁的任务释放后，其他任务才能重新访问该公共资源，此时互斥锁再次上锁，如此确保同一时刻只有一个任务正在访问这个公共资源，保证了公共资源操作的完整性。





实现互斥锁功能

cmsis_os2的API互斥锁接口简介:

接口名	功能描述
osMutexNew	创建互斥锁
osMutexAcquire	获取互斥锁
osMutexRelease	释放互斥锁
osMutexDelete	删除互斥锁

创建互斥锁: `osMutexNew (const osMutexAttr_t *attr);`

获取互斥锁: `osMutexAcquire (osMutexId_t mutex_id, uint32_t timeout);`

释放互斥锁: `osMutexRelease (osMutexId_t mutex_id);`

删除互斥锁: `osMutexDelete (osMutexId_t mutex_id);`



互斥锁扩展实验

扩展实验代码

```
void HighPrioThread(void)
{
    // wait 1s until start actual work
    osDelay(1000U);
    osStatus_t status;
    while (1)
    {
        // try to acquire mutex
        status = osMutexAcquire(mutex_id, osWaitForever);
        if(status != osOK)
        {
            printf("acquire mutex failed\r\n");
        }
        else
        {
            printf("acquire mutex success\r\n");
        }

        printf("HighPrioThread is running.\r\n");
        osDelay(3000U);

        status = osMutexRelease(mutex_id);
        if(status != osOK)
        {
            printf("release mutex failed\r\n");
        }
        else
        {
            printf("release mutex success\r\n");
        }
    }
}
```

```
void LowPrioThread(void)
{
    osStatus_t status;
    while (1)
    {
        status = osMutexAcquire(mutex_id, osWaitForever);
        printf("LowPrioThread is running.\r\n");
        if(status != osOK)
        {
            printf("acquire mutex failed\r\n");
        }
        else
        {
            printf("acquire mutex success\r\n");
        }
        // block mutex for 3s
        osDelay(3000U);
        status = osMutexRelease(mutex_id);
        if(status != osOK)
        {
            printf("release mutex failed\r\n");
        }
        else
        {
            printf("release mutex success\r\n");
        }
    }
}
```

```
status = osMutexDelete(mutex_id);
if(status != osOK)
{
    printf("delete mutex failed\r\n");
}
else
{
    printf("delete mutex success\r\n");
}
status = osMutexDelete(mutex_id);
if(status != osOK)
{
    printf("delete mutex failed\r\n");
}
else
{
    printf("delete mutex success\r\n");
}
```

本节小结

- 1、了解互斥锁的概念
- 2、掌握如何创建和删除互斥锁
- 3、掌握如何利用互斥锁实现对共享资源的独占式处理



谢谢观看

开源从小熊派开始

OPEN-SOURCE STARTED WITH THE BEARPI