凌智電子
LOCKZHINER Electronic

OpenHarmonyOS内核开发

互斥锁

凌智電子
LOCKZHINER Electronic

目 录

CONTENTS

01 什么是互斥锁

02 互斥锁的相应接口

03 如何创建、申请、释放互斥锁

凌智電子
LOCKZHINER Electronic

01 什么是互斥锁

- 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。
- 任意时刻互斥锁的状态只有两种，开锁或闭锁。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。当该任务释放它时，该互斥锁被开锁，任务失去该互斥锁的所有权。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。
- 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。

02 互斥锁的接口

互斥锁接口的头文件

/kernel/liteos_m/kernel/include/los_mux.h

OpenHarmonyOS内核开发中，互斥锁接口有很多，主要分为几大类：

（1）创建和删除互斥锁；

（2）申请互斥锁；

（3）释放互斥锁；

02 互斥锁的接口

功能分类	接口名	功能描述
互斥锁的创建和删除	LOS_MuxCreate	创建互斥锁
	LOS_MuxDelete	删除指定的互斥锁
互斥锁的申请和释放	LOS_MuxPend	申请指定的互斥锁
	LOS_MuxPost	释放指定的互斥锁

02 互斥锁的接口

UINT32 LOS_MuxCreate(UINT32 *muxHandle);

该函数主要功能是创建互斥锁。参数muxHandle为互斥锁的句柄。

UINT32 LOS_MuxPend(UINT32 muxHandle, UINT32 timeout);

该函数主要功能是等待一定时间申请指定的互斥锁。参数muxHandle为互斥锁的句柄，timeout为等待时间。

UINT32 LOS_MuxPost(UINT32 muxHandle);

该函数主要功能是释放指定互斥锁。参数muxHandle为互斥锁的句柄。

03 如何创建、申请、释放互斥锁

打开sdk下面路径的文件

vendor/lockzhiner/rk2206/samples/a4_kernel_mutex/kernel_mutex_example.c

在mutex_example函数中，通过LOS_MuxCreate函数创建了互斥锁，并创建的两个线程write_thread和read_thread。

```
void mutex_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;
    ret = LOS_MuxCreate(&m_mutex_id);
    if (ret != LOS_OK)
    {
        printf("Falied to create Mutex\n");
    }
```

如何创建、申请、释放互斥锁

```
    task1.pfnTaskEntry = (TSK_ENTRY_FUNC)write_thread;
    task1.uwStackSize = 2048;
    task1.pcName = "write_thread";
    task1.usTaskPrio = 24;
    ret = LOS_TaskCreate(&thread_id1, &task1);
    if (ret != LOS_OK)
    {
        printf("Falied to create write_thread ret:0x%x\n", ret);
        return;
    }
```

```
    task2.pfnTaskEntry = (TSK_ENTRY_FUNC)read_thread;
    task2.uwStackSize = 2048;
    task2.pcName = "read_thread";
    task2.usTaskPrio = 25;
    ret = LOS_TaskCreate(&thread_id2, &task2);
    if (ret != LOS_OK)
    {
        printf("Falied to create read_thread ret:0x%x\n", ret);
        return;
    }
}
```

如何创建、申请、释放互斥锁

在write_thread线程函数中，先获得互斥锁，写入数据，并持有它时延迟3s。

```
void write_thread()
{
    while (1)
    {
        LOS_MuxPend(m_mutex_id, LOS_WAIT_FOREVER);
        m_data++;
        printf("write_thread write data:%u\n", m_data);
        LOS_Msleep(3000);
        LOS_MuxPost(m_mutex_id);
    }
}
```

在read_thread线程函数中，延时1s后，申请互斥锁等待，线程被阻塞；3S后write_thread线程释放互斥锁，read_thread线程获得互斥锁，读取数据。

```
void read_thread()
{
    /*delay 1s*/
    LOS_Msleep(1000);
    while (1)
    {
        LOS_MuxPend(m_mutex_id, LOS_WAIT_FOREVER);
        printf("read_thread read data:%u\n", m_data);
        LOS_Msleep(1000);
        LOS_MuxPost(m_mutex_id);
    }
}
```

如何创建、申请、释放互斥锁

4.修改编译脚本

修改 vendor/lockzhiner/rk2206/sample 路径下 BUILD.gn 文件，指定 a4_kernel_mutex 参与编译。

```
"./a4_kernel_mutex:mutex_example",
```

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

```
hardware_LIBS = -lhal_iothardware -lhardware -lmutex_example
```

5.编译固件

```
hb set -root .
hb set
hb build -f
```

如何创建、申请、释放互斥锁

6.烧写固件

7.通过串口查看结果通过串口查看结果write_thread线程函数先写入数据阻塞3s，read_thread线程函数阻塞3s后读取数据。

运行结果

```
write_thread write data:1
read_thread read data:1
write_thread write data:2
read_thread read data:2
write_thread write data:3
read_thread read data:3
……
```

凌智電子
LOCKZHINER Electronic

谢谢聆听

单击此处添加副标题内容