10.7 互斥信号量

为了避免优先级反转这个问题，UCOSIII 支持一种特殊的二进制信号量：互斥信号量，用它可以解决优先级反转问题，如图 10.7.1 所示。

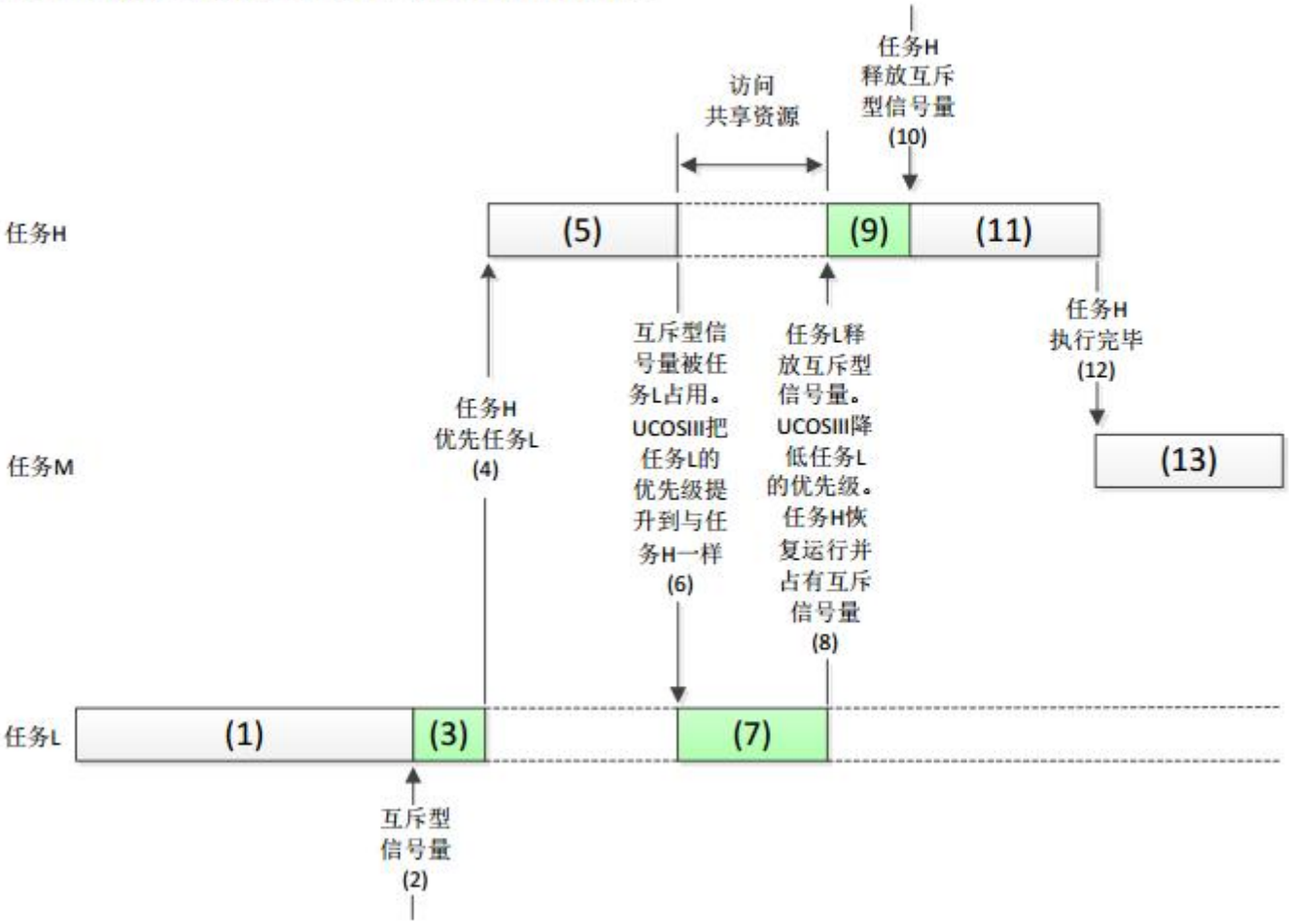

图 10.7.1 使用互斥型信号量访问共享资源

(1) 任务 H 与任务 M 处于挂起状态，等待某一事件的发生，任务 L 正在运行中。

(2) 某一时刻任务 L 想要访问共享资源，在此之前它必须先获得对应资源的互斥型信号量。

(3) 任务 L 获得互斥型信号量并开始使用该共享资源。

(4) 由于任务 H 优先级高，它等待的事件发生后便剥夺了任务 L 的 CPU 使用权。

(5) 任务 H 开始运行。

(6) 任务 H 运行过程中也要使用任务 L 在使用的资源，考虑到任务 L 正在占用着资源，UCOSIII 会将任务 L 的优先级升至同任务 H 一样，使得任务 L 能继续执行而不被其他中等优先级的任务打断。

(7) 任务 L 以任务 H 的优先级继续运行，注意此时任务 H 并没有运行，因为任务 H 在等待任务 L 释放掉互斥信号量。

(8) 任务 L 完成所有的任务，并释放掉互斥型信号量，UCOSIII 会自动将任务 L 的优先级恢复到提升之前的值，然后 UCOSIII 会将互斥型信号量给正在等待着的任务 H。

(9) 任务 H 获得互斥信号量开始执行。

(10) 任务 H 不再需要访问共享资源，于是释放掉互斥型信号量。

(11) 由于没有更高优先级的任务需要执行，所以任务 H 继续执行。

(12) 任务 H 完成所有工作，并等待某一事件发生，此时 UCOSIII 开始运行在任务 H 或者任务 L 运行过程中已经就绪的任务 M。

(13) 任务 M 继续执行。

注意！只有任务才能使用互斥信号量(中断服务程序则不可以)，UCOSIII 允许用户嵌套使用互斥型信号量，一旦一个任务获得了一个互斥型信号量，则该任务最多可以对该互斥型信号量嵌套使用 250 次，当然该任务只有释放相同的次数才能真正释放这个互斥型信号量。

与普通信号量一样，对于互斥信号量也可以进行许多操作，如表 10.7.1 所示，文件 os_mutex.c 是关于互斥信号量的。

函数	描述
OSMutexCreate()	创建一个互斥信号量
OSMutexDel()	删除一个互斥型信号量
OSMutexPend()	等待一个互斥型信号量
OSMutexPendAbort()	取消等待
OSMutexPost()	释放一个互斥型信号量

表 10.7.1 互斥型信号量操作 API 函数

10.7.1 创建互斥型信号量

创建互斥信号量使用函数 OSMutexCreate()，函数原型如下：

```
void  OSMutexCreate (OS_MUTEX   *p_mutex,
                     CPU_CHAR   *p_name,
                     OS_ERR     *p_err)
```

p_mutex： 指向互斥型信号量控制块。互斥型信号量必须有用户应用程序进行实际分配，可以使用如下所示代码。

```
OS_MUTEX  MyMutex;
```

p_name： 互斥信号量的名字

p_err： 调用此函数后返回的错误码。

10.7.2 请求互斥型信号量

当一个任务需要对资源进行独占式访问的时候就可以使用函数 OSMutexPend()，如果该互斥信号量正在被其他的任务使用，那么 UCOSIII 就会将请求这个互斥信号量的任务放置在这个互斥信号量的等待表中。任务会一直等待，直到这个互斥信号量被释放掉，或者设定的超时时间到达为止。如果在设定的超时时间到达之前信号量被释放，UCOSIII 将会恢复所有等待这个信号量的任务中优先级最高的任务。

注意！如果占用该互斥信号量的任务比当前申请该互斥信号量的任务优先级低的话，OSMutexPend()函数会将占用该互斥信号量的任务的优先级提升到和当前申请任务的优先级一样。当占用该互斥信号量的任务释放掉该互斥信号量以后，恢复到之前的优先级。OSMutexPend()函数原型如下：

```
void  OSMutexPend (OS_MUTEX  *p_mutex,
                   OS_TICK   timeout,
                   OS_OPT    opt,
                   CPU_TS    *p_ts,
                   OS_ERR    *p_err)
```

p_mutex： 指向互斥信号量。

timeout： 指定等待互斥信号量的超时时间（时钟节拍数），如果在指定的时间内互斥信号量没有释放，则允许任务恢复执行。该值设置为 0 的话，表示任务将会一直等待下去，直到信号量被释放掉。

opt： 用于选择是否使用阻塞模式。

OS_OPT_PEND_BLOCKING 指定互斥信号量被占用时，任务挂起等待该互斥信号量。

OS_OPT_PEND_NON_BLOCKING 指定当互斥信号量被占用时，直接返回任务。

注意！当设置为 OS_OPT_PEND_NON_BLOCKING，是 timeout 参数就没有意义了，应该设置为 0。

p_ts： 指向一个时间戳，记录发送、终止或删除互斥信号量的时刻。

p_err： 用于保存调用此函数后返回的错误码。

10.7.3 发送互斥信号量

我们可以通过调用函数 OSMutexPost()来释放互斥型信号量，只有之前调用过函数 OSMutexPend()获取互斥信号量，才需要调用 OSMutexPost()函数来释放这个互斥信号量，函数原型如下：

```
void  OSMutexPost (OS_MUTEX  *p_mutex,
                   OS_OPT    opt,
                   OS_ERR    *p_err)
```

p_mutex: 指向互斥信号量。

opt: 用来指定是否进行任务调度操作

OS_OPT_POST_NONE 不指定特定的选项

OS_OPT_POST_NO_SCHED 禁止在本函数内执行任务调度操作。

p_err: 用来保存调用此函数返回的错误码。