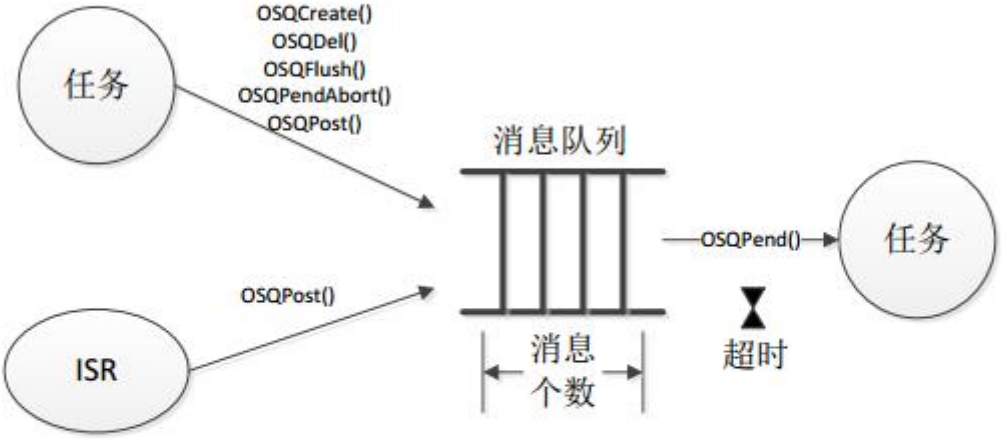
11.1 消息队列

消息一般包含：指向数据的指针，表明数据长度的变量和记录消息发布时刻的时间戳，指针指向的可以是一块数据区或者甚至是一个函数，消息的内容必须一直保持可见性，因为发布数据采用的是引用传递是指针传递而不是值传递，也就说，发布的数据本身不产生数据拷贝。

在 UCOSII 中有消息邮箱和消息队列，但是在 UCOSIII 中只有消息队列。消息队列是由用户创建的内核对象，数量不限制，图 11.1.1 展示了用户可以对消息队列进行的操作。



如图 11.1.1 消息队列相关操作

从图 11.1.1 中可以看出，中断服务程序只能使用 OSQPost()函数！在 UCOSIII 中对于消息队列的读取既可以采用先进先出(FIFO)的方式，也可以采用后进先出(LIFO)的方式。当任务或者中断服务程序需要向任务发送一条紧急消息时 LIFO 的机制就非常有用了。采用后进先出的方式，发布的消息会绕过其他所有的已经位于消息队列中的消息而最先传递给任务。

图 11.1.1 中接收消息的任务旁边的小沙漏表示任务可以指定一个超时时间，如果任务在这段时间内没有接收到消息的话就会唤醒任务，并且返回一个错误码告诉 UCOSIII 超时，任务是因为接收消息超时而被唤醒的，不是因为接收到了消息。如果将这个超时时间指定为 0 的话，那么任务就会一直等待下去，直到接收到消息。

消息队列中有一个列表，记录了所有正在等待获得消息的任务，如图 11.1.2 所示为多个任务可以在一个消息队列中等待，当一则消息被发布到队列中时，最高优先级的等待任务将获得该消息，发布方也可以向消息队列中所有等待的任务广播一则消息。

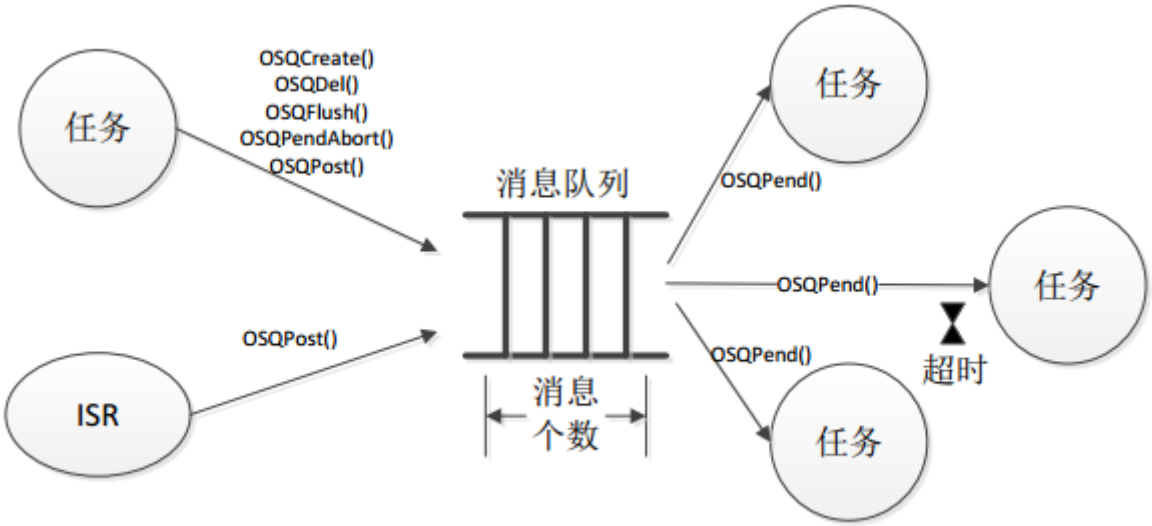


图 11.1.2 多个任务在等待一个消息队列

11.2 消息队列相关函数

有关消息队列的 API 函数如下表 11.2.1 所示。

函数	描述
OSQCreate()	创建一个消息队列
OSQDel()	删除一个消息队列
OSQFlush()	清空一个消息队列
OSQPend()	等待消息队列
OSQPendAbort()	取消等待消息队列
OSQPost()	向消息队列发送一条消息

我们常用的关于消息队列的函数其实只有三个，创建消息队列函数 OSQCreate()，向消息队列发送消息函数 OSQPost()和等待消息队列函数 OSQPend()。

11.2.1 创建消息队列

OSQCreate()函数用来创建一个消息队列，消息队列使得任务或者中断服务程序可以向一个或者多个任务发送消息，函数原型如下。

```
void    OSQCreate (OS_Q            *p_q,
                   CPU_CHAR        *p_name,
                   OS_MSG_QTY      max_qty,
                   OS_ERR          *p_err)
```

p_q： 指向一个消息队列，消息队列的存储空间必须由应用程序分配，我们采用如下的语句定义一个消息队列。

```
OS_Q   Msg_Que;
```

p_name： 消息队列的名字。

max_qty： 指定消息队列的长度，必须大于 0。当然，如果 OS_MSGs 缓冲池中没有足够多的 OS_MSGs 可用，那么发送消息将会失败，并且返回相应的错误码，指明当前没有 可用的 OS_MSGs

p_err： 保存调用此函数后返回的错误码

11.2.2 等待消息队列

当一个任务想要从消息队列中接收一个消息的话就需要使用函数 OSQPend()。当任务调用这个函数的时候，如果消息队列中有至少一个消息时，这些消息就会返回给函数调用者。函数原型如下：

```
void    *OSQPend (OS_Q             *p_q,
                  OS_TICK          timeout,
                  OS_OPT           opt,
                  OS_MSG_SIZE      *p_msg_size,
                  CPU_TS           *p_ts,
                  OS_ERR           *p_err)
```

p_q： 指向一个消息队列。

timeout： 等待消息的超时时间，如果在指定的时间没有接收到消息的话，任务就会被唤醒，接着运行。这个参数也可以设置为 0，表示任务将一直等待下去，直到接收到消息。

opt： 用来选择是否使用阻塞模式，有两个选项可以选择。

OS_OPT_PEND_BLOCKING 如果没有任何消息存在的话就阻塞任务，一直等待，直到接收到消息。

OS_OPT_PEND_NON_BLOCKING 如果消息队列没有任何消息的话任务就直接返回。

p_msg_size: 指向一个变量用来表示接收到的消息长度(字节数)。

p_ts: 指向一个时间戳，表明什么时候接收到消息。如果这个指针被赋值为NULL 的话，说明用户没有要求时间戳。

p_err: 用来保存调用此函数后返回的错误码。

如果消息队列中没有任何消息，并且参数 opt 为 OS_OPT_PEND_NON_BLOCKING 时，那么调用 OSQPend()函数的任务就会被挂起，直到接收到消息或者超时。如果有消息发送给消息队列，但是同时有多个任务在等待这个消息，那么 UCOSIII 将恢复等待中的最高优先级的任务。

11.2.3 向消息队列发送消息

可以通过函数 OSQPost()向消息队列发送消息，如果消息队列是满的，则函数 OSQPost()就会立刻返回，并且返回一个特定的错误代码，函数原型如下：

```
void  OSQPost (OS_Q          *p_q,
               void          *p_void,
               OS_MSG_SIZE   msg_size,
               OS_OPT        opt,
               OS_ERR        *p_err)
```

如果有多个任务在等待消息队列的话，那么优先级最高的任务将获得这个消息。如果等待消息的任务优先级比发送消息的任务优先级高，则系统会执行任务调度，等待消息的任务立即恢复运行，而发送消息的任务被挂起。可以通过 opt 设置消息队列是 FIFO 还是 LIFO。

如果有多个任务在等待消息队列的消息，则 OSQPost()函数可以设置仅将消息发送给等待任务中优先级最高的任务(opt 设置为 OS_OPT_POST_FIF 或者 OS_OPT_POST_LIFO)，也可以将消息发送给所有等待的任务(opt 设置为 OS_OPT_POST_ALL)。如果 opt 设置为 OS_OPT_POST_NO_SCHED，则在发送完消息后，会进行任务调度。

p_q: 指向一个消息队列。

p_void: 指向实际发送的内容，p_void 是一个执行 void 类型的指针，其具体含义由用户程序的决定。

msg_size: 设定消息的大小，单位为字节数。

opt: 用来选择消息发送操作的类型，基本的类型可以有下面四种。

OS_OPT_POST_ALL 将消息发送给所有等待该消息队列的任务，需要

OS_OPT_POST_LIFO 配合使用。

OS_OPT_POST_FIFO	待发送消息保存在消息队列的末尾
OS_OPT_POST_LIFO	待发送的消息保存在消息队列的开头
OS_OPT_POST_NO_SCHED	禁止在本函数内执行任务调度。

我们可以使用上面四种基本类型来组合出其他几种类型，如下：

OS_OPT_POST_FIFO + OS_OPT_POST_ALL

OS_OPT_POST_LIFO + OS_OPT_POST_ALL

OS_OPT_POST_FIFO + OS_OPT_POST_NO_SCHED

OS_OPT_POST_LIFO + OS_OPT_POST_NO_SCHED

OS_OPT_POST_FIFO + OS_OPT_POST_ALL + OS_OPT_POST_NO_SCHED

OS_OPT_POST_LIFO + OS_OPT_POST_ALL + OS_OPT_POST_NO_SCHED

p_err： 用来保存调用此函数后返回的错误码。