## Swoole版本：1.7.5-beta

版本更换通知：从第七章开始，我把分析的源码版本升到了1.7.5-beta。在已经分析过的代码中存在了一些变动，但没有到会影响理解的地步，所以再次不再更新前面的章节。如果确认前面的章节有重大变更需要重新分析的，我会给出声明。

Swoole内部封装了Linux提供的msg queue用于提供消息队列的操作。消息队列的声明在swoole.h文件的360 – 378 行。声明如下：

typedef struct \_swQueue\_Data

{

long mtype; /\* type of received/sent message \*/

char mdata[sizeof(swEventData)]; /\* text of the message \*/

} swQueue\_data;

typedef struct \_swQueue

{

void \*object;

int blocking;

int (\*in)(struct \_swQueue \*, swQueue\_data \*in, int data\_length);

int (\*out)(struct \_swQueue \*, swQueue\_data \*out, int buffer\_length);

void (\*free)(struct \_swQueue \*);

int (\*notify)(struct \_swQueue \*);

int (\*wait)(struct \_swQueue \*);

} swQueue;

int swQueueMsg\_create(swQueue \*p, int wait, int msg\_key, long type);

第一个结构体swQueue\_Data封装了消息队列中的消息，其中 mtype 用于标记该消息是需要发送的消息还是从队列中读出的消息，mdata用于存放消息内容（swEventData结构体将在下一章解析）。

最后一行的函数用于创建一个消息队列，第二个参数wait指定阻塞类型，msg\_key指定创建的消息队列的键值，type标记消息队列读数据的类型

type=0，接受该队列的第一个消息，并将它返回给调用者

type>0，接受类型type的第一个消息

type<0，接受小于等于type绝对值的最低类型的第一个消息

swQueueMsg\_create函数的具体声明在Msg.c文件中，其核心代码如下：

swQueueMsg \*object = sw\_malloc(sizeof(swQueueMsg));

if (object == NULL)

{

return -1;

}

if (blocking == 0)

{

object->ipc\_wait = IPC\_NOWAIT;

}

else

{

object->ipc\_wait = 0;

}

p->blocking = blocking;

msg\_id = msgget(msg\_key, IPC\_CREAT | O\_EXCL | 0666);

源码解释：申请内存，分配消息队列的结构体内存，并判定消息队列的阻塞类型。接着调用msgget函数创建消息队列。

这里说明msgget第三个参数中的几个值的意义。IPC\_CREAT表明，若不存在以key为索引的消息队列，则创建新的队列；否则，返回已创建的消息队列的id。O\_EXCL表示若打开一个已存在的消息队列，则返回-1。0666为该消息队列的读写权限。

swQueueMsg有对应的三个操作函数，分别是：

int swQueueMsg\_in(swQueue \*p, swQueue\_data \*in, int data\_length);

int swQueueMsg\_out(swQueue \*p, swQueue\_data \*out, int buffer\_length);

void swQueueMsg\_free(swQueue \*p);

这三个函数用于向队列中写入消息、从队列中读出消息、释放队列。

1. swQueueMsg\_in函数循环调用msgsnd直到消息发送成功或无法再次发送。
2. swQueueMsg\_out函数调用msgrcv函数接收指定类型的消息
3. swQueueMsg\_free函数调用msgctl释放一个消息队列，IPC\_RMID参数代表立即释放全部资源。