**static int ecryptfs\_acquire\_free\_msg\_ctx(struct ecryptfs\_msg\_ctx \*\*msg\_ctx)**

从ecryptfs\_msg\_ctx\_free\_list链表上取下一个空闲的ecryptfs\_msg\_ctx。

**static void ecryptfs\_msg\_ctx\_free\_to\_alloc(struct ecryptfs\_msg\_ctx \*msg\_ctx)**

将msg\_ctx从ecryptfs\_msg\_ctx\_free\_list移到ecryptfs\_msg\_ctx\_alloc\_list。

**void ecryptfs\_msg\_ctx\_alloc\_to\_free(struct ecryptfs\_msg\_ctx \*msg\_ctx)**

**int ecryptfs\_find\_daemon\_by\_euid(struct ecryptfs\_daemon \*\*daemon, uid\_t euid,**

**struct user\_namespace \*user\_ns)**

在ecryptfs\_daemon\_hash哈希表中查找对应euid和user\_ns的ecryptfs\_daemon进程。

**int ecryptfs\_spawn\_daemon(struct ecryptfs\_daemon \*\*daemon, uid\_t euid,**

**struct user\_namespace \*user\_ns, struct pid \*pid)**

根据对应euid和user\_ns创建新的daemon并通过daemon的euid\_chain成员链入ecryptfs\_daemon\_hash表。

**int ecryptfs\_exorcise\_daemon(struct ecryptfs\_daemon \*daemon)**

释放daemon。

**int ecryptfs\_process\_quit(uid\_t euid, struct user\_namespace \*user\_ns, struct pid \*pid)**

根据euid和user\_ns在ecryptfs\_daemon\_hash找到并释放相应的daemon。

**int ecryptfs\_process\_response(struct ecryptfs\_message \*msg, uid\_t euid,**

**struct user\_namespace \*user\_ns, struct pid \*pid,u32 seq)**

根据传进来的msg->index，在ecryptfs\_msg\_ctx\_arr消息队列中找到对应的消息msg\_ctx，进一步找到daemon，作一系列一致性判断后将msg复制到msg\_ctx的msg。一致性判断如下：

|  |  |
| --- | --- |
| ctx\_euid = task\_euid(msg\_ctx->task) | euid |
| (msg\_ctx->task)->user->user\_ns | user\_ns |
| daemon->pid | pid |
| msg\_ctx->state | ECRYPTFS\_MSG\_CTX\_STATE\_PENDING |
| msg\_ctx->counter | seq |

**static int**

**ecryptfs\_send\_message\_locked(char \*data, int data\_len, u8 msg\_type,struct ecryptfs\_msg\_ctx \*\*msg\_ctx)**

根据current\_euid()和current\_user\_ns()找到daemon，调用：

ecryptfs\_send\_miscdev(data, data\_len, \*msg\_ctx, msg\_type, 0,daemon)；

**int ecryptfs\_send\_message(char \*data, int data\_len,struct ecryptfs\_msg\_ctx \*\*msg\_ctx)**

对ecryptfs\_send\_message\_locked的封装。

**int ecryptfs\_wait\_for\_response(struct ecryptfs\_msg\_ctx \*msg\_ctx,struct ecryptfs\_message \*\*msg)**

这里有一个等待msg\_ctx->state状态为ECRYPTFS\_MSG\_CTX\_STATE\_DONE的死循环。

上述消息准备就绪后，\*msg = msg\_ctx->msg，然后将msg\_ctx放入空闲队列供再次使用。

**int \_\_init ecryptfs\_init\_messaging(void)**

初始化ecryptfs\_daemon\_hash，大小由ecryptfs\_number\_of\_users决定，缺省值为4；

初始化全局消息队列ecryptfs\_msg\_ctx\_arr，大小由ecryptfs\_message\_buf\_len决定，缺省值为32，同时将每个msg\_ctx元素链接到ecryptfs\_msg\_ctx\_free\_list链表上。

最后调ecryptfs\_init\_ecryptfs\_miscdev注册ecryptfs文件系统的misc设备。

**void ecryptfs\_release\_messaging(void)**