kucomms kernel programmers guide

Table of Contents

Introduction

In order for a kernel module to communicate with a userspace application, it is necessary to define three callback functions and then register those callback functions with the kucomms module.

Defining and registering callbacks

The first step is to declare the callback functions. The functions shown below have no implementation and are examples only.

```
static bool
my_message_hlr(const struct Message * message,
               MessageQueueHeaderPtr tx_msgq,
               const __u64 rx_msgq_queueLength,
               const __u64 tx_msgq_queueLength,
               void * userData)
{
        return true;
}

static bool
my_work_hlr(void * userData)
{
        return false;
}

static void
my_timer_hlr(const __u64 time, void * userData)
{
}
```

The next step is to register the callbacks with the kucomms module.

The user must register in the module init function and must unregister in the module exit function.

```
const char * devname = "kucomms_myname";

static int __init init_mymodule(void)
{
        bool ok = kucomms_register(
                devname,
                strlen(devname),
                my_message_hlr,
                my_work_hlr,
                my_timer_hlr,
                0);

        if (!ok) return -ENODEV;

        return 0;
}

static void __exit exit_mymodule(void)
{
        kucomms_unregister_wait(devname, strlen(devname));
}

module_init(init_mymodule);
module_exit(exit_mymodule);
```

Sending a message

The x function is used to send a message.

Below are some examples of sending a message.

```
static bool
my_message_hlr(const struct Message * message,
               MessageQueueHeaderPtr tx_msgq,
               const __u64 rx_msgq_queueLength,
               const __u64 tx_msgq_queueLength,
               void * userData)
{
        struct kucomms_file_data * pfd = (struct kucomms_file_data *)userData;

        // Send message received back to the sender.
        bool ok = message_queue_add_tx0(pfd, message);

        return true;
}

static bool
my_work_hlr(void * userData)
{
        struct kucomms_file_data * pfd = (struct kucomms_file_data *)userData;
        struct Message * message;
        __u64 dataLength;
        bool ok;

        dataLength = 10; // build a message
        message = vmalloc(message_get_message_length(dataLength));

        message->m_length = dataLength;
        message->m_type = 0;
        message->m_id = 0;
        message->m_userValue = 0;
        for (__u32 u0=0; u0<dataLength; u0++) message->m_data[u0] = 0;

        ok = message_queue_add_tx0(pfd, message); // send message

        vfree(message);

        return false;
}
```