kucomms userspace programmers guide

Table of Contents

Introduction

In order to create a userspace application that can communicate with a kernel module, it is necessary to define three callback functions and then run the main loop of the message endpoint.

Defining and registering callbacks

The first step is to define 3 classes, the user callbacks will be members of these classes.

```
#include "MessageManager.h"

class MyMessageHandler : public MessageHandler
{
public:
        bool hlr(const struct Message * message,
                MessageQueueWriter & tx_msgq,
                std::vector<MessageQueueWriter> & tx_msgq_list);
};

class MyWorkHandler : public WorkHandler
{
public:
        bool hlr(std::vector<MessageQueueWriter> & tx_msgq_list);
};

class MyTimerHandler : public TimerHandler
{
public:
        void hlr(const __u64 time,
                std::vector<MessageQueueWriter> & tx_msgq_list);
};
```

The next step is to declare the handler methods. The methods shown below have no implementation and are examples only.

```
bool
MyMessageHandler::hlr(const struct Message * message,
                MessageQueueWriter & tx_msgq,
                std::vector<MessageQueueWriter> & tx_msgq_list)
{
        return true;
}

bool
MyWorkHandler::hlr(std::vector<MessageQueueWriter> & tx_msgq_list)
{
        return false;
}

void
MyTimerHandler::hlr(const __u64 time,
                std::vector<MessageQueueWriter> & tx_msgq_list)
{
}
```

The last step is to run the main loop of the message endpoint.

```
static bool g_stopped = false;

void
terminate_signal_hanlder(int sig)
{
        g_stopped = true;
}

int main(int argc, char ** argv)
{
        signal(SIGTERM, terminate_signal_hanlder);

        MyMessageHandler msghlr;
        MyWorkHandler workhlr;
        MyTimerHandler timerhlr;

        bool ok = MessageManager::run(
                                "/dev/kucomms_myname"
                                1024*1024,
                                g_stopped,
                                msghlr,
                                workhlr,
                                timerhlr);

        return 0;
}
```