Санкт-Петербургский Государственный Университет

Математико-механический факультет

Кафедра системного программирования

Организация надежных соединений через виртуальные каналы

Курсовая работа студента 345 группы

Бурдун Федора Викторовича

Научный руководитель
аспирант кафедры
системного программирования

А. В. Бондарев

Санкт-Петербург

2011

Содержание

Введение

Задача разработки новой операционной системы включает в себя создание надежной устойчивой системы, где каждая ее часть играет строго определенную роль. Для повышения надежности отдельных узлов этой системы вводятся заранее оговоренные интерфейсы взаимодействия и проектируются эти модули исходя из уже строго поставленной задачи. Так же обдуманная унификация интерфейсов и функциональности модуля позволяет использовать его в достаточно широких областях.

В реализации Операционных систем порой присутствует необходимость иметь возможность с одним физическим интерфейсом связывать несколько виртуальных, примерами данной необходимости являются: реализация текстового терминала с несколькими экземплярами консолей на которых выполняются различные оболочки, передача данных разным сервисам операционной системы по одному физическому каналу данных, (управление несколькими параметрами и передача команд роботу по bluetooth), как в принципе и любое другое взаимоотношение с несколькими сервисами операционной системы в рамках одного канала связи.

Правильно спроектировав данный модуль мы имеем возможность на любом промежуточном этапе обработки данных воспользоваться другими модулями предоставляющими функциональность шифрования, контроля целостности либо сжатия потока данных на лету.

Так, например, реализация защищенного соединения bluetooth в рамках подобного подхода к проектированию, сводится к написанию собственно сплиттера, который будет взаимодействовать с драйвером, модуля шифрования, и модуля подсчета-коррекции ошибок.

Постановка задачи

Задачей данной курсовой является выработка подхода и рассмотрение различных вариантов реализации так называемого "сплиттера".

Чтобы апробировать данную возможность было предложено реализовать модуль в контексте всем известного текстового терминала. Таким образом предполагалось создать модуль который сможет предоставлять интерфейс связывающий пару физических или логических устройств (одно для ввода, другое для вывода данных) с несколькими обработчиками, так чтобы это можно было использовать не только для запуска shell-ов но и для передачи/интерпретации прямых команд непосредственного управления, передачи шифрованных данных, прозрачно для прикладных программистов ОС.

Существующие решения

Схожая к поставленной задаче решается в ОС “linux” через драйвера построенные над IP либо bluetooth. Таким образом возможность шифрования и контроля целостности (если это не предусмотрено заранее драйвером) остается возможной только на прикладном уровне, что не всегда удовлетворяет поставленным требованиям.

В какой-то степени часть задачи, а именно взаимодействие с различными сервисами в рамках нашей ОС по одному каналу данных, в случае сетевого устройства (например ethernet) может решаться построением стека протоколов таких как tcp/ip или подобных. Минусом данного решения является очевидно невозможность использовать шифрование на самом низком уровне, а так же громоздкость и избыточность данного решения, что в свою очередь увеличивает время отклика и перегружает и без того не многочисленные ресурсы встраиваемых систем.

Так же возможно решение при использовании существующих драйверов терминала, но минусы подобного подхода аналогичны описанным выше.

Таким образом в какой-то степени нам приходится заниматься проектированием с нуля, в случае которого неизбежны ошибки, возможно даже в изначальном подходе. Именно по этому, для минимизации подобных неприятных ситуаций, на начальном этапе было решено ограничится реализацией только драйвера терминала с возможностью промежуточной обработки данных, и лишь затем переносить (или возможно опять таки исходя из полученного опыта перепроектировать) полученное решение на задачу коммуникации виртуальных интерфейсов.

Реализация

В рамках данной курсовой работы была написана совокупность модулей вместе выполняющие роль текстового терминала работающего в каноническом режиме (что в свою очередь означает, что простейшая обработка вводимой строки: удаление символа, слова, перемещение курсора, выполняется на уровне драйвера терминала).

На ниже представленных рисунках мы можем видеть результат работы драйвера. С парой устройств ввода/вывода было связано несколько консолей, которые сами являются устройствами ввода/вывода для программной оболочки esh (embox-shell). Оболочки выполняются в отдельных потоках, что так же можно видеть на рисунках 1 и 2.

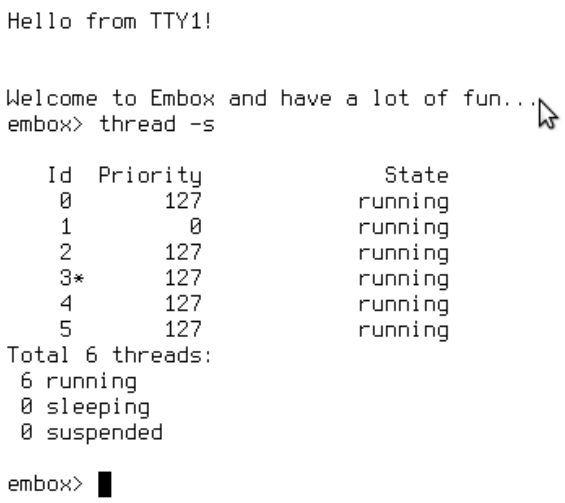

Рис. 1

После переключения, посредством нажатия <Ctrl>+<F2> мы можем видеть другую консоль и так же выполнять на ней любые команды.

Рис. 2

```
Hello from TTY2!


Welcome to Embox and have a lot of fun...
embox> thread
Usage: thread [-h] [-s] [-k <thread_id>]
thread: Command returned with code -1: Not permitted

embox> thread -s

   Id  Priority             State
    0       127           running
    1         0           running
    2       127           running
    3       127           running
    4*      127           running
    5       127           running
Total 6 threads:
        6 running
        0 sleeping
        0 suspended

embox>
```

Структура получившегося драйвера следующая (рис. 3):

1. I/O device driver

драйвер символьного устройства реализующий интерфейс: putchar, getchar, т.е. предоставляющий возможность читать/писать символ в физическое либо виртуальное устройство.

2. SPLITTER

модуль перенаправляющий поток данных от I/O device к console и обратно, служит своеобразным переключателем, который имеет возможность дополнительной обработки данных на любом этапе. Переключение может осуществляться по произвольному событию либо наличию специального маркера в потоке данных. В случае терминала подобным маркером является escape-последовательность кодирующая комбинацию клавиш <Ctrl>+<Fn>

3. console

драйвер консоли в простейшем случае так же реализует интерфейс putchar, getchar упомянутый выше, но при необходимости может реализовывать методы позиционирования

курсора, прокрутки экрана, изменение параметров выводимого текста (цвет текста, фона, флаг мерцания)

4. shell

Командный интерпретатор (либо как его еще называют командная оболочка) собственно модуль отвечающий за выполнение вводимых команд. Так в примере выше мы пользовались командой ядра ОС “embox” `thread -s` позволяющей узнать некоторую информацию о текущих нитях исполнения в запущенной операционной системе.

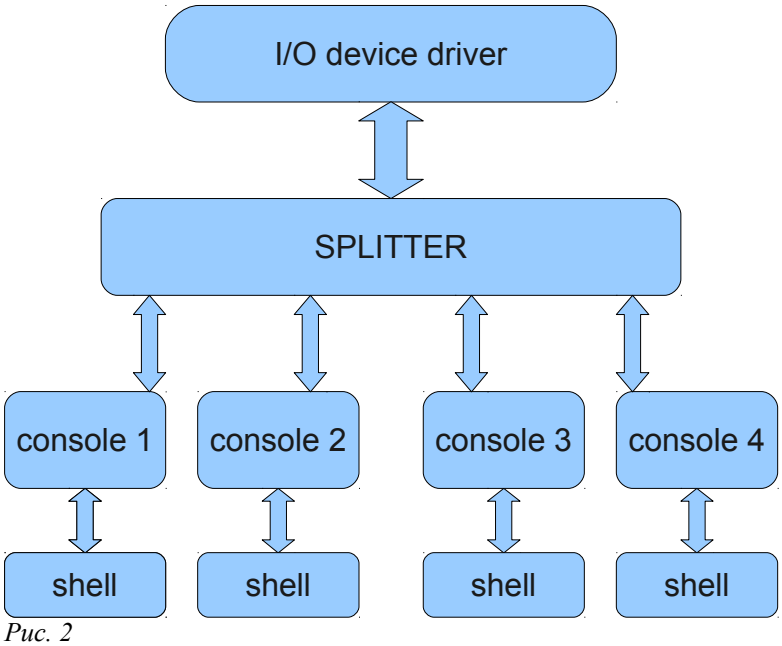

Рис. 2

В действительности embox-shell взаимодействует с консолью не напрямую, а через вызовы функции библиотеки readline. Библиотека реализует интерфейс командной строки и ее редактирование, так, что программисту пользующемуся библиотекой не приходится задумываться о режиме работы терминала (канонический/не канонический). Так же в ней может быть реализована работа с историей команд и авто-дополнение.

Использование данной библиотеки позволяет быть реализации оболочки практически тривиальной, что заметно упрощает ее отладку и уменьшает количество мест для потенциальных ошибок. Еще одним плюсом такого разделения функциональности является легкое последующее написание интерактивных приложений в рамках данной ОС.

Заключение

Таким образом результатом данной курсовой работы является написание совокупности модулей, которые уже сейчас активно используются в ОС “embox” в качестве полнофункционального текстового терминала. Часть модулей изначально проектировались как переиспользуемые (такие как SPLITTER, библиотека readline) и в дальнейшем будут использоваться и для создания интерактивных пользовательских приложений, а так же для создания различных типов соединений, например безопасного соединения bluetooth.

Следующим естественным этапом работы в проекте будет написание модулей отвечающих за шифрование и контроль целостности, а так же их применение в рамках уже существующих протоколов взаимодействия.

За успехами проекта вы можете следить на сайте http://code.google.com/p/embox

Мой личный в клад в проект может быть оценен по коммитам пользователя *ignwah* на данном сайте.

Список используемой литературы

[1] Операционные системы: разработка и реализация. / Э. Танненбаум, А. Вудхалл

[2] Design and Evaluation of an Embedded Real-time Micro-kernel / Kuljeet Singh

[3] Combining Physical and Virtual Channels. / Charles Steinfield, Harry Bouwman, Thomas Adelaar

[4] Improving the Performance of Real-Time Communication Services / J. Fern´ ndez, J. M. Garc´a

[5] RFC1193. Client requirements for real-time communication services / D. Ferrari