Курсовая работа
на тему:
“Оптимизация потоков для операционной системы Embox”.

Мальковский Николай Владимирович, 345 гр.

Санкт-Петербург, 2011 г.

Оглавление.

Введение.

Возможность выполнять несколько задач одновременно - одна из основных функций, которые должна предоставлять операционная система. Эту задачу решает механизм потоков(иногда их называют нитями). Общая принцип действия этого механизма заключается в том, что при выполнении некоторой задачи у нас будут два условия перехода к другому заданию - во-первых, когда мы выполнили текущую задачу, а во-вторых, некоторый временной предел: если мы уже довольно долго выполняли текущую задачу, то следует исполнить другие задачи, а потом вернуться к этой.

В самом общем случае этот механизм можно реализовать следующим образом:

- Каждой задаче предоставляется “поток”, в каждый фиксированный момент времени существует биекция между задачами и потоками
- Есть специальный объект, так называемый *планировщик*, который решает когда следует остановить выполнение текущего потока и какой поток поставить следующим на выполнение

Таким образом, если нужно поддерживать многозадачность, то от того, насколько хорошо реализован этот механизм, будет зависеть производительность системы в целом, поэтому очень важно, чтобы такие операции как переключение, добавление и удаление потоков выполнялись как можно быстрее. В основном оптимизации потоков заключаются в уменьшении времени обработки перечисленных операций, а также памяти расходуемых на описание одного потока. В этой работе будут рассмотрены некоторые такие оптимизации.

Постановка задачи

Рассмотрим такую модель некоторой системы:

- В каждый момент времени система может находиться в одном из известных состояний, всего состояний конечное число
- Есть конечный, заранее определенный набор команд, с помощью которых можно воздействовать на систему
- Для любой пары <состояние, команда> определена пара <состояние, действие>
- При поступлении очередной команды система переходит из текущего состояние в состояние, определенное в предыдущем пункте, сопровождая его соответствующим действием

Таким образом поведение системы можно описать конечным автоматом, состояния которого - состояния системы, входной алфавит - возможные команды, выходной алфавит - возможные действия.

Предполагается, что действия не являются дискретными, т.е. на их выполнение должно уйти некоторое время. Таким образом управление системой будет происходить следующим образом:

- При поступлении очередной команды, она сразу возбуждает переход в следующее состояние и соответствующее действие, иначе записывается в очередь.
- После того, как система закончила очередное действие, следует проверить, нету ли в очереди команды на исполнение. Если есть, то исполнить ее, иначе ждать, пока не поступит.

Стоит отметить, что в такой постановке, у нас нету возможности отменить команду(т.е. остановить выполнение соответствующего ему действия, если оно выполняется в данный момент).

Реализация.

Основной особенностью такой модели является факт, что нам вполне хватает одного потока, чтобы управлять ею. Один из плюсов совершенно очевиден - при наличие только одного потока операция переключения потоков (и соответственно дорогого переключения контекста) полностью отсутствует. Чтобы добиться такого результата, обработку команд следует производить с помощью аппаратных прерываний. То есть для каждой команды выделяется прерывание. Обработка каждого такого прерывания будет заключаться в том, чтобы сообщить планировщику о том, что пришла очередная команда, который в свою очередь будет регулировать работу рабочего потока.

Теперь опишем сказанное выше более формально и объединим с абстрактной моделью, описанной ранее:

1. Каждой команде соответствует свое аппаратное прерывание
2. При получении очередной команды происходит соответствующее аппаратное прерывание, которое сообщает планировщику о себе.
3. Планировщик получает управление в двух случаях: после аппаратного прерывания и по завершении обработки очередной команды потоком.
4. Планировщик при получении управления после аппаратного прерывания смотрит, занят ли рабочий поток. Если да, то

добавляет в очередь сообщение о том, что была получена соответствующая команда, дает эту команду на исполнение потоку.

5. При получении управления после завершения обработки очередной команды потоком, планировщик смотрит, есть ли в очереди команда. Если есть, то поток начинает ее исполнять, иначе планировщик переводит поток в спящий режим до появления следующего прерывания.
6. И наконец, поток просто выполняет то, что ему сказали. Выполнил - отчитался.

Итак, осталось только обсудить, каким образом будут общаться между собой планировщик и поток. Предлагается делать это с помощью простой модели событий. Само событий представляет собой пару <handler, data> - обработчик и информация о событии.

Для каждой команды заведем определенной событие. Обработчиком(handler) для события будет собственно то действие, соответствующее команде, соответствующей этому событию.

Далее у нас есть две процедуры:
event_send(....) и event_dispatch(void).

Первая добавляет сообщение в очередь(само сообщение - параметр процедуры). Вторая ничего не делает, если поток занят, иначе берет первое в очереди сообщение и ставит потоку на исполнение ее обработчик. Если очередь пуста, то переводит поток в спящий режим.

При возникновении прерывания вызывается event_send() с параметрами, прописанными в этом прерывании, и event_dispatch(). После завершения обработки очередной команды (в данном случае - сообщения) поток вызывает event_dispatch().

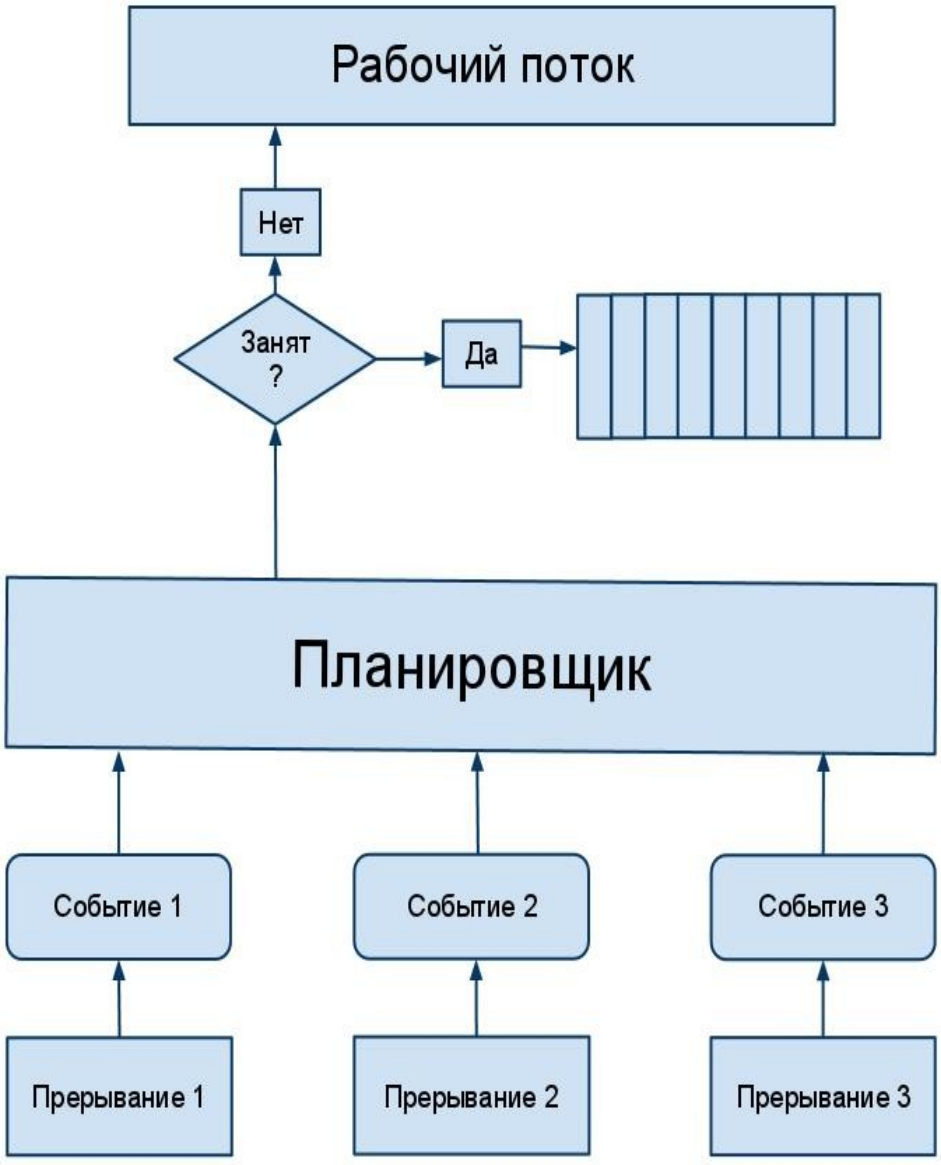

Рис. 1

Рис. 1 показывает обработку команды, Рис. 2 показывает случай, когда поток заканчивает выполнение очередной функции.

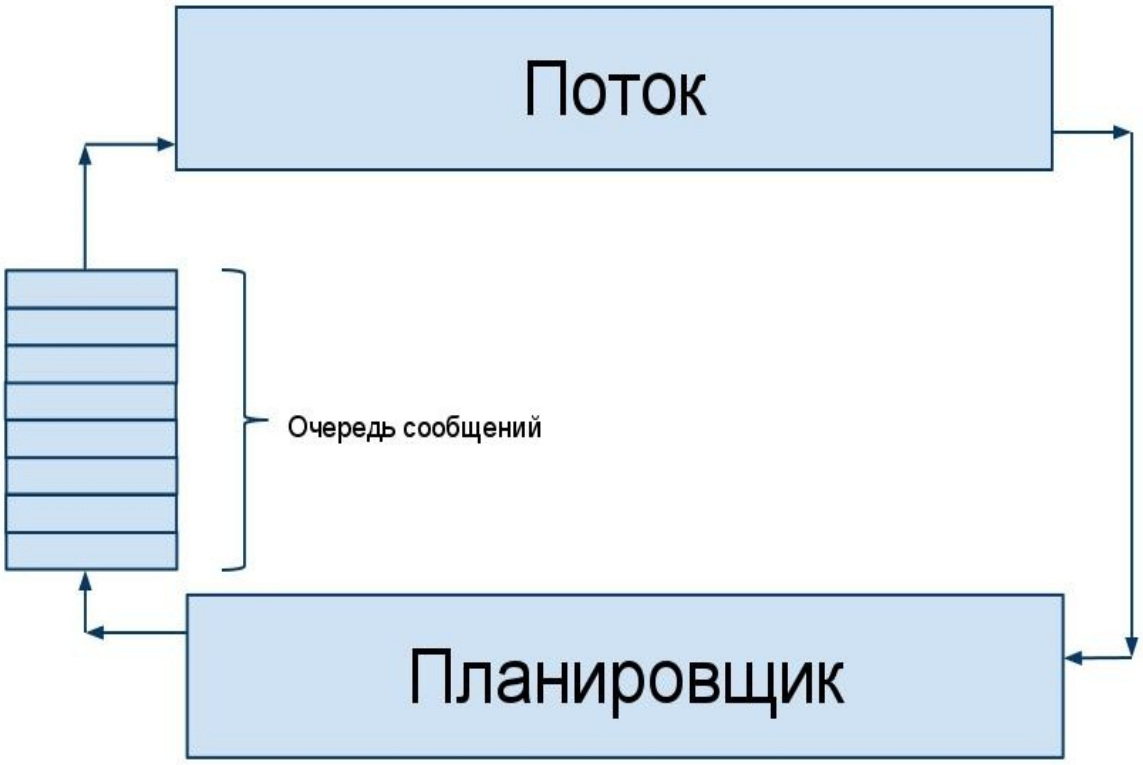

Рис. 2

Итоги.

Перечислим плюсы и минусы получившейся модели и ее реализации.

Плюсы:

- Расходы на содержания потоков малы: одно единственное добавление потока происходит при загрузке системы, больше ни переключений, ни добавление/удалений происходить не будет
- Быстрый отклик на команды
- Отсутствие простоев в работе системы - система ничего не делает только если все полученные команды уже были обработаны и других не поступало

Минусы:

- Нет возможности отменить команду