Библиотека libevent для асинхронного неблокирующего ввода/вывода: Часть 1. Общий обзор

Алексей Снастин
независимый разработчик ПО
начальник отдела

14.07.2011

Первая статья цикла, посвященного библиотеке libevent, предназначенной для обработки оповещений о событиях и организации асинхронного ввода/вывода, описывает общую структуру библиотеки.

В этом цикле статей рассматривается библиотека libevent, предназначенная для обработки оповещений о событиях и организации асинхронного ввода/вывода. В первой статье описывается общая структура библиотеки. Вторая статья посвящена основам использования libevent: наибольшее внимание уделено обработке событий. В третьей статье речь идёт о механизмах буферизации ввода/вывода. Тема четвёртой, заключительной статьи - применение libevent в сетевых приложениях.

Введение

Библиотека libevent написана Нильсом Провосом (Niels Provos) для упрощения и унификации поддержки механизма асинхронного неблокирующего ввода/вывода и механизма оповещения о событиях с помощью выполнения обратного вызова (callback) функции при наступлении заданного события для дескриптора файла или при достижении заданного таймаута (timeout). Библиотека является свободно распространяемой на условиях лицензии BSD. Получить пакет для установки libevent можно из репозиториев соответствующего дистрибутива Linux, а исходные тексты доступны на авторском сайте. На момент написания данной статьи текущей версией является 2.0.10-stable.

> Версии библиотеки libevent до 1.4.7 считаются полностью устаревшими.
>
> Версии до 1.3e вообще классифицируются как "абсолютно неприменимые на практике".

В настоящее время libevent поддерживает механизмы select(2), kqueue(2), poll(2), epoll(4) и /dev/poll. В пока ещё "нештатном режиме" имеется экспериментальная поддержка обработки сигналов в реальном времени. API-интерфейсы библиотеки абсолютно независимы от внутренних механизмов обработки событий и буферизации, поэтому при обновлении libevent, как правило, не требуется заново переписывать использующие её приложения.

Таким образом, библиотека обеспечивает переносимость приложений и позволяет выбрать наиболее эффективный и гибкий механизм оповещения о событиях из всех доступных в целевой операционной системе. Кроме того, libevent может применяться для создания многопоточных (multi-threaded) программ. Скомпилировать и собрать библиотеку libevent можно в операционных системах Linux, *BSD, Mac OS X, Solaris и Windows.

1. Структура библиотеки

Бибилиотека libevent организована таким образом, чтобы обеспечить работу написанных на её основе программ на всех поддерживаемых платформах наиболее эффективным образом и предоставить разработчику удобный, не перегруженный лишними подробностями API-интерфейс.

1.1. Логическая структура - модули

Главные модули, основные "рабочие лошадки", - это event и event_base. Они обеспечивают абстрактный API-интерфейс для разнообразных внутренних механизмов неблокирующего ввода/вывода, основанных на отслеживании событий, зависимых от конкретной платформы. Это позволяет разработчику узнавать, когда сокеты готовы к чтению или записи, реализовать базовую функциональность механизма таймаута и отлавливать сигналы (события) операционной системы.

Функции модуля bufferevent предоставляют более удобную обёртку вокруг основанного на отслеживании событий ядра библиотеки libevent, позволяют приложению организовать операции буферизованного чтения и записи и не просто сообщают о факте готовности сокетов к той или иной операции, а позволяют точно узнать, когда в действительности выполняется ввод/вывод.

Модуль evbuffer содержит реализации буферов, являющихся основой для bufferevent-событий, и предоставляет все необходимые функции для работы с этими буферами.

Независимость от деталей реализации работы в сетевой среде на различных платформах, то есть выделение обобщённой функциональности обеспечивает модуль evutil.

Кроме того, имеются ещё три компонента: evhttp - упрощённая реализация HTTP-сервера и HTTP-клиента, evdns - упрощённая реализация клиента/сервера DNS и evrpc - упрощённая реализация механизма удалённого вызова процедур (RPC).

1.2. Файловая структура - библиотеки

По умолчанию в составе libevent устанавливаются следующие библиотеки:

- libevent_core - общее ядро для обеспечения функциональности механизмов обработки событий и буферизации. Эта библиотека содержит все функции модулей event_base, bufferevent, evbuffer и evutil.
- libevent_extra - эта библиотека определяет дополнительную функциональность, зависящую от конкретного сетевого протокола, которая не всегда необходима для того или иного приложения. Сюда включена поддержка протоколов HTTP, DNS и RPC.

- libevent - устаревающая библиотека, существующая лишь для обеспечения обратной совместимости; в неё входит содержимое и libevent_core, и libevent_extra. В создаваемых заново приложениях использовать эту библиотеку не рекомендуется; возможно, библиотечный файл libevent будет полностью исключён уже из следующей версии библиотеки.

Установка следующих библиотечных файлов зависит от текущей используемой платформы:

libevent_pthreads добавляет поддержку многопоточности и механизмов блокировки на основе переносимой библиотеки pthreads. Этот библиотечный файл отделён от ядра libevent_core для того, чтобы без затруднений отключать многопоточность в тех случаях, когда она не нужна.

Библиотечный файл libevent_openssl обеспечивает поддержку зашифрованного обмена данными с помощью библиотеки OpenSSL и также отделён от libevent_core. Поэтому если в приложении не планируется использование зашифрованных соединений, то при сборке этот файл можно отключить.

1.3. Заголовочные файлы (headers)

Все доступные заголовочные файлы библиотеки libevent устанавливаются в подкаталог event2 и делятся на три основные категории:

Заголовочные файлы API определяют текущие внешние открытые интерфейсы к функциям библиотеки libevent. Эти заголовочные файлы не имеют каких-либо специальных суффиксов в основном имени.

Заголовочные файлы для обеспечения обратной совместимости включают определения для устаревших и не рекомендуемых к применению функций. Эти файлы не следует использовать за исключением того случая, когда выполняется переход программы со старой версии libevent на одну из последних новых версий.

> В библиотеке libevent версии 2.0 и более поздних старые заголовочные файлы продолжают существовать в виде обёрток для новых заголовочных файлов.

Заголовочные файлы структур - в этих файлах содержатся определения разнообразных внутренних структур, необходимых для функционирования библиотеки libevent. Прямое обращение к любой такой структуре возможно, но при этом необходимо помнить о потенциальной опасности такого действия: оно может нарушить бинарную совместимость программы с другими версиями libevent, а подобного рода нарушения зачастую весьма трудно поддаются отладке. У заголовочных файлов структур в основном имени имеется суффикс "*_struct.h".

2. Базовые принципы и механизмы

Одним из несомненных достоинств библиотеки libevent является простота и удобство её практического применения. Каждая программа, использующая libevent, должна включать заголовочный файл:

```
#include <event2/event.h>
```

а при компоновке выполняемого файла необходимо задать для линкера флаг -levent.

В исходном коде приложения требуется инициализация libevent, предшествующая использованию какой бы то ни было функции из этой библиотеки. Для этого вызывается либо функция event_init(), либо функция event_base_new().

2.1. Оповещение о событии

Для каждого дескриптора файла, за которым необходимо вести наблюдение, определяется структура события (тип event). Чтобы организовать оповещение, нужно добавить эту структуру в список отслеживаемых событий с помощью вызова event_add(). Структура данного события физически существует в выделенном для неё фрагменте памяти всё то время, в течение которого она продолжает оставаться активной. После создания структуры и внесения её в список вызывается dispatch-функция для инициализации цикла и наблюдения за событиями, включёнными в список.

2.2. Простой пример обработки событий

Этот пример иллюстрирует описанную выше последовательность действий и практическое применение механизмов библиотеки libevent. Приведённый в Листинге 1 фрагмент следит за определённым событием и при получении оповещения о его наступлении просто завершает работу.

Листинг 1. Обработка единственного ожидаемого события

```
#include <event2/event.h>

/* Функция обратного вызова, в которой выполняется прерывание цикла ожидания */
void call_back( int sock, short what, void *arg )
{
  struct event_base *base = arg;
  event_base_loopbreak( base );
}

void main_loop( struct event_base *base, evutil_socket_t watch_fd )
{
  struct event *watch_event;

  /* Создание нового события, генерируемого при выполнении операции чтения
   * из отслеживаемого сокета (файла). При наступлении этого события
   * вызывается функция обратного вызова call_back, которая выполняет
   * немедленный выход из цикла ожидания данного события и
   * завершение работы программы.
   */
  watch_event = event_new( base, watch_fd, EV_READ, call_back, base );
  event_add( watch_event, NULL );
  event_base_dispatch( base );
}
```

2.3. Буферы ввода/вывода

Библиотека libevent предлагает дополнительный уровень абстракции, размещённый выше обычного механизма обратных вызовов, реагирующих на заданные события, - так

называемое "буферизованное событие" (buffered event). Такое буферизованное событие наряду с штатными атрибутами предоставляет ещё и буферы ввода и вывода, которые заполняются и очищаются автоматически. Таким образом, разработчик освобождается от необходимости работать напрямую с потоками ввода/вывода, вместо этого он считывает данные из буфера ввода и записывает данные в буфер вывода.

Для работы с буферами создаётся структура типа bufferevent, которая инициализируется функцией bufferevent_new(), после чего может использоваться многократно с возможностью её подключения bufferevent_enable() и отключения bufferevent_disable(). Операция чтения из буфера выполняется с помощью функции bufferevent_read(), и, соответственно, операция записи в буфер - bufferevent_write().

Если операция чтения разрешена, то объект bufferevent будет пытаться считать данные из заданного сокета (дескриптора файла) и выполнить обратный вызов read. Обратный вызов write выполняется в том случае, когда буфер вывода освобождён до такого состояния, при котором размер данных, хранящихся в буфере, становится меньше определённого порогового значения (low watermark). По умолчанию это пороговое значение принято равным 0.

2.4. Таймеры

Библиотеку libevent можно использовать для создания таймеров, то есть, объектов, отсчитывающих заданный интервал времени, по истечении которого выполняется определённый обратный вызов. Таймер создаётся и инициализируется из обычного события event с помощью функции evtimer_set(). Включить (активизировать) таймер можно функцией evtimer_add(), а отключить - с помощью функции evtimer_del().

2.5. Механизм таймаутов

Для файловых дескрипторов не так важны события таймеров, как события таймаутов, которые генерируются, если на отслеживаемом дескрипторе файла отсутствует какая бы то ни было активность в течение заданного интервала времени. Структура event превращается в триггер таймаута с помощью функции timeout_set(). Само собой триггер необходимо "взвести" - для этого служит функция timeout_add(). Чтобы "разрядить" триггер и отменить событие таймаута, необходимо воспользоваться функцией timeout_del().

2.6. HTTP-сервер, управляемый событиями

В любое приложение, которое использует библиотеку libevent, можно без особых затруднений встроить простейший HTTP-сервер, управляемый событиями. Для этого необходимо дополнительно включить в программу заголовочный файл evhttp.h. Функция evhttp_new() создаёт сервер. Функция evhttp_bind_socket() добавляет пары "адрес-порт" для ожидания запросов. Для запросов, поступающих на определённые URI, можно зарегистрировать собственный обратный вызов с помощью функции evhttp_set_cb(). Обработку запросов на "неопределённые" URI (для которых не было зарегистрировано отдельных обратных вызовов) можно передать обобщённому обратному вызову через функцию evhttp_set_gencb().

2.7. Организация удалённого вызова процедур

Включение в программу заголовочного файла evrpc.h позволяет организовать рабочую среду для выполнения и обработки удалённых вызовов процедур (RPC). Операции маршалинга (marshaling) и демаршалинга (unmarshaling) любых структур данных берёт на себя в полном объёме библиотека libevent.

2.8. Сервис DNS

Функции асинхронного DNS-резолвера (resolver) становятся доступными разработчику при включении в программу заголовочного файла evdns.h. Последовательность операций по созданию и инициализации аналогична действиям при использовании других сервисов. Для преобразования символьного имени хоста в IP-адрес вызывается функция evdns_resolve_ipv4(), а для выполнения обратного поиска и преобразования следует обратиться к функции evdns_resolve_reverse(). Разумеется, здесь также используется механизм обратных вызовов для того, чтобы избежать взаимоблокировок при выполнении операций поиска.

Заключение

Библиотека libevent представляет собой удобное и эффективное инструментальное средство, работающее на нескольких платформах и обеспечивающее быстрое и качественное создание приложений, в которых требуется асинхронный неблокирующий ввод/вывод.

В данной статье была описана общая структура библиотеки. Вторая статья посвящена основам использования libevent: основное внимание уделено обработке событий. В третьей статье речь идёт о механизмах буферизации ввода/вывода. Тема четвёртой, заключительной статьи - применение libevent в сетевых приложениях.

Об авторе

Алексей Снастин

Алексей Снастин - независимый разработчик ПО, консультант и переводчик с английского языка технической и учебной литературы по ИТ. Принимал участие в разработке сетевых офисных приложений типа клиент/сервер на языке C в среде Linux.