ВикипедиЯ

Событийно-ориентированное программирование

Материал из Википедии — свободной энциклопедии

Событи́йно-ориенти́рованное программи́рование (англ. *event-driven programming*; в дальнейшем СОП) — парадигма программирования, в которой выполнение программы определяется событиями — действиями пользователя (клавиатура, мышь), сообщениями других программ и потоков, событиями операционной системы (например, поступлением сетевого пакета).

СОП можно также определить как способ построения компьютерной программы, при котором в коде (как правило, в головной функции программы) явным образом выделяется *главный цикл приложения*, тело которого состоит из двух частей: *выборки события* и *обработки события*.

Как правило, в реальных задачах оказывается недопустимым длительное выполнение обработчика события, поскольку при этом программа не может реагировать на другие события. В связи с этим при написании событийно-ориентированных программ часто применяют автоматное программирование.

Содержание

Сфера применения

Событийно-ориентированное программирование, как правило, применяется в трёх случаях:

1. при построении пользовательских интерфейсов (в том числе графических);
2. при создании серверных приложений в случае, если по тем или иным причинам нежелательно порождение обслуживающих процессов;
3. при программировании игр, в которых осуществляется управление множеством объектов.

Применение в серверных приложениях

Событийно-ориентированное программирование применяется в серверных приложениях для решения проблемы масштабирования на 10000 одновременных соединений и более.

В серверах, построенных по модели «один поток на соединение», проблемы с масштабируемостью возникают по следующим причинам:

- слишком велики накладные расходы на структуры данных операционной системы, необходимые для описания одной задачи (сегмент состояния задачи, стек);
- слишком велики накладные расходы на переключение контекстов.

Философской предпосылкой для отказа от потоковой модели серверов может служить высказывание Алана Кокса: «Компьютер — это конечный автомат. Потоковое программирование нужно тем, кто не умеет программировать конечные автоматы»[1].

Серверное приложение при событийно-ориентированном программировании реализуется на системном вызове, получающем события одновременно от многих дескрипторов (мультиплексирование). При обработке событий используются исключительно неблокирующие операции ввода-вывода, чтобы ни один дескриптор не препятствовал обработке событий от других дескрипторов.

Мультиплексирование

Для мультиплексирования соединений могут быть использованы следующие средства операционной системы:

- select (большинство UNIX систем). Плохо масштабируется, из-за того, что список дескрипторов представлен в виде битовой карты;
- poll и epoll (Linux);
- kqueue (FreeBSD);
- /dev/poll (Solaris);
- IO completion port (Windows);
- POSIX AIO на текущий момент только для операций дискового ввода-вывода;
- io submit и eventfd для операций дискового ввода-вывода.

Примеры реализаций

- Веб-серверы:
  - Node.js
  - nginx
  - lighttpd
  - Tornado
- Прокси-серверы:
  - Squid

Применение в настольных приложениях

В современных языках программирования события и обработчики событий являются центральным звеном реализации графического интерфейса пользователя. Рассмотрим, к примеру, взаимодействие программы с событиями от мыши. Нажатие правой клавиши мыши вызывает системное прерывание, запускающее определённую процедуру внутри операционной системы. В этой процедуре происходит поиск окна, находящегося под курсором мыши. Если окно найдено, то данное событие посылается в очередь обработки сообщений этого окна. Далее, в зависимости от типа окна, могут генерироваться дополнительные события. Например, если окно является кнопкой (в Windows все графические элементы являются окнами), то дополнительно генерируется событие нажатия на кнопку. Отличие последнего события в том, что оно более абстрактно, а именно, не содержит координат курсора, а говорит просто о том, что было произведено нажатие на данную кнопку.

Обработчик события может выглядеть следующим образом (на примере C#):

```
    private void button1_Click(object sender, EventArgs e)
    {
        MessageBox.Show("Была нажата кнопка");
    }
```

Здесь обработчик события представляет собой процедуру, в которую передается параметр sender, как правило содержащий указатель на источник события. Это позволяет использовать одну и ту же процедуру для обработки событий от нескольких кнопок, различая их по этому параметру.

Языки программирования

В языке C# события реализованы как элемент языка и являются членами классов. Механизм событий здесь реализует шаблон проектирования Publisher/Subscriber. Пример объявления события:

```
public class MyClass
{
    public event EventHandler MyEvent;
}
```

Здесь *EventHandler* — делегат, определяющий тип процедуры обработчика событий. Подписка на событие производится следующим образом:

```
        myClass.MyEvent += new EventHandler(Handler);
```

Здесь *myClass* — экземпляр класса *MyClass*, *Handler* — процедура-обработчик. Событие может иметь неограниченное количество обработчиков. При добавлении обработчика события он добавляется в специальный стек, а при возникновении события вызываются все обработчики по их порядку в стеке. Отписка от события, то есть удаление обработчика производится аналогично, но с использованием оператора «-=».

Разные языки программирования поддерживают СОП в разной степени. Наиболее полной поддержкой событий обладают следующие языки (неполный список):

- Perl (события и демоны `DAEMON`, и их приоритеты `PRIO`),[2]
- Delphi (язык программирования),
- ActionScript 3.0,
- C# (события `event`)[3],
- JavaScript (действия пользователя).

Остальные языки, в большей их части, поддерживают события как обработку исключительных ситуаций.

Инструменты и библиотеки

- Node.js, событийно-ориентированный I/O фреймворк на JavaScript движке V8
- Cocoa & Objective-C, рефлексивный объектно-ориентированный язык программирования, добавляющий сообщения в стиле Smalltalk в язык Си.
- GLib
- **Gui4Cli**[4], событийно-ориентированный язык программирования для Windows
- libsigc++
- libevent
- POCO
- **libasync**, часть библиотек sfs и sfslite[5], эффективная событийная библиотека для C++
- Perl Object Environment
- AnyEvent, EV — модули на Perl для событийно-ориентированного программироания
- PRADO, компонентный событийно-ориентированный инструмент для Web-программирования на PHP 5
- Tcl

- Twisted, Python
- Qt, кроссплатформенная библиотека виджетов для C++, основанная на модели управления событиями. Существует сокращённая версия, называемая Qt/Console, из которой исключён код поддержки виджетов, и представляющий собой управляемый событиями фреймворк, в который также включены некоторые дополнительные средства, вроде кроссплатформенной работы с сетью, многопоточности и работы с XML.
- **QP** — семейство открытых событийно-ориентированных окружений для встроенных систем реального времени[6]
- Simple Unix Events a.k.a. **SUE**[7], простая объектно-ориентированная библиотека для построения событийно-ориентированных программ под Unix на языке C++.

См. также

- Автоматное программирование
- Callback (программирование)

Англоязычные источники

- описание (http://c2.com/cgi/wiki?EventDrivenProgramming) из Portland Pattern Repository (http://c2.com/ppr/)
- Event-Driven Programming: Introduction, Tutorial, History (http://eventdrivenpgm.sourceforge.net/) — учебное пособие Стефана Ферга (Stephen Ferg)
- Event Driven Programming (http://www.freenetpages.co.uk/hp/alan.gauld/tutevent.htm) учебное пособие Алана Голда (Alan Gauld)
- Martin Fowler. *Event Collaboration* (http://www.martinfowler.com/eaaDev/EventCollaboration.html)
- Ben Watson. Transitioning from Structured to Event-Driven Programming (http://www.devhood.com/tutorials/tutorial_details.aspx?tutorial_id=504)
- Jonathan Simon. *Rethinking Swing Threading* (http://today.java.net/lpt/a/32)
- Chris McDonald. *The event driven programming style* (http://www.csse.uwa.edu.au/cnet/eventdriven.html)
- Christopher Diggins. *Event Driven Programming using Template Specialization* (http://codeproject.com/cpp/static-callbacks.asp)
- Stefan Schiffer and Joachim Hans Fröhlich. *Concepts and Architecture of Vista — a Multiparadigm Programming Environment* (http://www.swe.uni-linz.ac.at/people/schiffer/se-94-17/se-94-17.htm)
- Event-Driven Programming and Agents (http://docs.eiffel.com/eiffelstudio/general/guided_tour/language/invitation-09.html)
- LabWindows/CVI Resources (http://zone.ni.com/devzone/devzone.nsf/webcategories/FCE7EA7ECA51169C862567A9005878EA)
- Comment (http://javalobby.org/forums/thread.jspa?threadID=13874&messageID=91806918&tstart=0) by Tim Boudreau
- Complex Event Processing and Service Oriented Architecture (http://news.tmcnet.com/news/2006/08/18/1816129.htm)
- Event-driven programming and SOA: Jack van Hoof. *How EDA extends SOA and why it is important;;* (http://soa-eda.blogspot.com/2006/11/how-eda-extends-soa-and-why-it-is.html)
- Пример с открытым кодом: Distributed Publish/Subscribe Event System (http://www.codeplex.com/pubsub)
- Событийно-ориентированное программирование на языке Java: Rex Youung. *Jsasb* (https://jsasb.dev.java.net/)

Материалы на русском

- *Н. Н. Непейвода.* 13. Лекция: Событийное программирование // Стили и методы программирования. *курс лекций. учебное пособие* (http://www.intuit.ru/department/se/progstyles/13/). — М.: Интернет-университет информационных технологий, 2005. — С. 213—222. — 316 с. — ISBN 5-9556-0023-X.
- *С.В. Зыков.* Лекции №15 и №16 // Введение в теорию программирования. Объектно-ориентированный подход (http://www.intuit.ru/department/se/tppobj/). — Интернет-университет информационных технологий.
- *О. В. Ануфриев.* О методике обучения основам событийного программирования (http://www.ict.edu.ru/vconf/files/3195.rtf)] (рус.). Новосибирский государственный педагогический университет. Проверено 29 октября 2010. Архивировано (http://www.webcitation.org/65tNjeXma) 3 марта 2012 года.
- А. П. Полищук, С. А. Семериков. *Программирование в X Window средствами Free Pascal* (http://fpc.by.ru/xwin/xwin.html)

Ссылки

1. Linux-Kernel Archive: Re: Alan Cox quote? (was: Re: accounting (http://www.uwsg.indiana.edu/hypermail/linux/kernel/0106.2/0405.html)
2. *Н. Н. Непейвода.* 13. Лекция: Событийное программирование // Стили и методы программирования. *курс лекций. учебное пособие* (http://www.intuit.ru/department/se/progstyles/13/). — М.: Интернет-университет информационных технологий, 2005. — С. 213—222. — 316 с. — ISBN 5-9556-0023-X.
3. *С.В. Зыков.* Лекции №15 и №16 // Введение в теорию программирования. Объектно-ориентированный подход (http://www.intuit.ru/department/se/tppobj/). — Интернет-университет информационных технологий.
4. Gui4Cli Home page (http://gui4cli.com)
5. sfslite: overview (http://www.okws.org/doku.php?id=sfslite:overview)
6. Download from Quantum Leaps (http://www.quantum-leaps.com/downloads)
7. The Simple Unix Events (SUE) library homepage (http://www.croco.net/software/sue/)