**Тезаурусный словарь**

Поток выполнения (тред; от англ. thread — нить) — наименьшая единица обработки, исполнение которой может быть назначено ядром операционной системы. Реализация потоков выполнения и процессов в разных операционных системах отличается друг от друга, но в большинстве случаев поток выполнения находится внутри процесса. Несколько потоков выполнения могут существовать в рамках одного и того же процесса и совместно использовать ресурсы, такие как память, тогда как процессы не разделяют этих ресурсов. В частности, потоки выполнения разделяют инструкции процесса (его код) и его контекст (значения переменных, которые они имеют в любой момент времени). В качестве аналогии потоки выполнения процесса можно уподобить нескольким вместе работающим поварам. Все они готовят одно блюдо, читают одну и ту же кулинарную книгу с одним и тем же рецептом и следуют его указаниям, причём не обязательно все они читают на одной и той же странице.

Многопоточность, как широко распространённая модель программирования и исполнения кода, позволяет нескольким потокам выполняться в рамках одного процесса. Эти потоки выполнения совместно используют ресурсы процесса, но могут работать и самостоятельно. Многопоточная модель программирования предоставляет разработчикам удобную абстракцию параллельного выполнения. Однако, пожалуй, наиболее интересное применение технологии имеется в том случае, когда она применяется к одному процессу, что позволяет его параллельное выполнение на многопроцессорной системе.

**Ментальная карта**

**Классическая модель потоков**

Многопоточный процесс

Аналогичный способ

# Классическая модель потоков

Разобравшись в пользе потоков и в порядке их использования, давайте рассмотрим их применение более пристально. Модель процесса основана на двух независимых понятиях: группировке ресурсов и выполнении. Иногда их полезно отделить друг от друга, и тут на первый план выходят потоки. Сначала будет рассмотрена классическая модель потоков, затем изучена модель потоков, используемая в Linux, которая размывает грань между процессами и потоками.

Согласно одному из взглядов на процесс, он является способом группировки в единое целое взаимосвязанных ресурсов. У процесса есть адресное пространство, содержащее текст программы и данные, а также другие ресурсы. Эти ресурсы могут включать открытые файлы, необработанные аварийные сигналы, обработчики сигналов, учетную информацию и т. д. Управление этими ресурсами можно значительно облегчить, если собрать их воедино в виде процесса.

Другое присущее процессу понятие — поток выполнения — обычно сокращается до слова поток. У потока есть счетчик команд, отслеживающий, какую очередную инструкцию нужно выполнять. У него есть регистры, в которых содержатся текущие рабочие переменные. У него есть стек с протоколом выполнения, содержащим по одному фрейму для каждой вызванной, но еще не возвратившей управление процедуры. Хотя поток может быть выполнен в рамках какого-нибудь процесса, сам поток и его процесс являются разными понятиями и должны рассматриваться по отдельности. Процессы используются для группировки ресурсов в единое образование, а потоки являются «сущностью», распределяемой для выполнения на центральном процессоре.

Потоки добавляют к модели процесса возможность реализации нескольких в значительной степени независимых друг от друга выполняемых задач в единой среде процесса. Наличие нескольких потоков, выполняемых параллельно в рамках одного процесса, является аналогией наличия нескольких процессов, выполняемых параллельно на одном компьютере. В первом случае потоки используют единое адресное пространство и другие ресурсы. А в последнем случае процессы используют общую физическую память, диски, принтеры и другие ресурсы. Поскольку потоки обладают некоторыми свойствами процессов, их иногда называют облегченными процессами. Термин «многопоточный режим» также используется для описания ситуации, при которой допускается работа нескольких потоков в одном и том же процессе. В главе 1 было показано, что некоторые центральные процессоры обладают непосредственной аппаратной поддержкой многопоточного режима и проводят переключение потоков за наносекунды.

На рис. 2.8, а показаны три традиционных процесса. У каждого из них имеется собственное адресное пространство и единственный поток управления. В отличие от этого, на рис. 2.8, б показан один процесс, имеющий три потока управления. Хотя в обоих случаях у нас имеется три потока, на рис. 2.8, а каждый из них работает в собственном адресном пространстве, а на рис. 2.8, б все три потока используют общее адресное пространство.

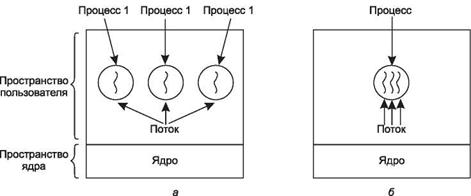


Рис. 2.8. а — три процесса, у каждого из которых по одному потоку; б — один процесс с тремя потоками

Когда многопоточный процесс выполняется на однопроцессорной системе, потоки выполняются, сменяя друг друга.

На рис. 2.1 мы видели работу процессов в многозадачном режиме. За счет переключения между несколькими процессами система создавала иллюзию параллельно работающих отдельных последовательных процессов. Многопоточный режим осуществляется аналогичным способом. Центральный процессор быстро переключается между потоками, создавая иллюзию, что потоки выполняются параллельно, пусть даже на более медленном центральном процессоре, чем реально используемый. При наличии в одном процессе трех потоков, ограниченных по скорости вычисления, будет казаться, что потоки выполняются параллельно и каждый из них выполняется на центральном процессоре, имеющем скорость, которая составляет одну треть от скорости реального процессора.

Различные потоки в процессе не обладают той независимостью, которая есть у различных процессов. У всех потоков абсолютно одно и то же адресное пространство, а значит, они так же совместно используют одни и те же глобальные переменные. Поскольку каждый поток может иметь доступ к любому адресу памяти в пределах адресного пространства процесса, один поток может считывать данные из стека другого потока, записывать туда свои данные и даже стирать оттуда данные. Защита между потоками отсутствует, потому что ее невозможно осуществить и в ней нет необходимости. В отличие от различных процессов, которые могут принадлежать различным пользователям и которые могут враждовать друг с другом, один процесс всегда принадлежит одному и тому же пользователю, который, по-видимому, и создал несколько потоков для их совместной работы, а не для вражды. В дополнение к использованию общего адресного пространства все потоки, как показано в табл. 2.4, могут совместно использовать одни и те же открытые файлы, дочерние процессы, ожидаемые и обычные сигналы и т. п. Поэтому структура, показанная на рис. 2.8, а, может использоваться, когда все три процесса фактически не зависят друг от друга, а структура, показанная на рис. 2.8, б, может применяться, когда три потока фактически являются частью одного и того же задания и активно и тесно сотрудничают друг с другом.