УЧИТЕЛЬ ГОДА РОССИИ 2019 ВСЕРОССИ

ОПЕРАЦИОННЫЕ СИСТЕМЫ. ПРОЦЕССЫ И ПОТОКИ: УЧЕБНОЕ ПОСОБИЕ

Автор/создатель: Илюшкин Б.И.

2

Голосов: 5

Пособие разработано в соответствии с государственными образовательными стандартами высшего профессионального образования по направлению подготовки дипломированного специалиста: 654600 - "Информатика и вычислительная техника"(специальность 220100 - "Вычислительные машины, комплексы, системы и сети") и направлению подготовки бакалавра 552800 - "Информатика и вычислительная техника". В пособии рассмотрены классификация и архитектура современных операционных систем, концепции управления процессами и потоками, а также механизмы управления виртуальной памятью. Учебное пособие предназначено для студентов, изучающих дисциплину "Операционные системы" в шестом семестре.

Приведенный ниже текст получен путем автоматического извлечения из оригинального PDF-документа и предназначен для предварительного просмотра.

Изображения (картинки, формулы, графики) отсутствуют.

Страницы ← предыдущая следующая →

1 2 3 4 5 6 7 8 9 10

Завер-

Новый шающийся

Готовый к выполнению

Выпол-
няющийся

Блокиро-
ванный

а) Схема перехода состояний

Очередь
готовых процессов Диспетчеризация Освобождение
Процессор

Пауза

Очередь блокированных Ожидание события
процессов

б) Схема с одной очередью блокированных процессов

Рис 2.3. Модель процесса с пятью состояниями

2.4. Потоки

Поток является последовательностью команд, обрабатываемых процессором. В рамках одного процесса могут находиться один или несколько потоков. Традиционный подход, при котором каждый процесс представляет собой единый поток выполнения, называется однопоточным [3]. Например, MS-DOS поддерживает один однопоточный пользовательский процесс. Некоторые ОС семейства UNIX поддерживают процессы множества пользователей, но при этом каждый из процессов содержит один поток. Многопоточностью (multithreading) называется способность ОС поддерживать в рамках одного процесса выполнение нескольких потоков. Примерами многопоточных систем являются среда выполнения Java, Windows 2000, Linux, Solaris и другие. На рис. 2.4 представлены варианты однопоточных и многопоточных процессов [3].

Рис 2.4. Однопоточные и многопоточные процессы

В многопоточной среде процесс можно рассматривать как структурную единицу объединения ресурсов и структурную единицу защиты. Ресурсами являются адресное пространство, открытые файлы, дочерние процессы, обработчики сигналов и многое другое. Под защитой подразумевается защищенный режим доступа к процессорам, другим процессам, файлам и ресурсам ввода-вывода. С другой стороны, процесс можно рассматривать как общий поток исполняемых команд, состоящий из нескольких отдельных потоков [5]. У каждого потока есть свой счетчик команд, регистры и стек. Таким образом, в многопоточной среде процессы используются для группирования ресурсов, а потоки являются объектами, поочередно выполняющимися на процессоре (в случае однопроцессорной вычислительной

системы), создавая впечатление параллельной работы потоков. Все потоки одного процесса разделяют между собой ресурсы процесса. Они находятся в общем адресном пространстве и имеют доступ к одним и тем же данным. Например, если один поток изменяет данные в памяти, то другие потоки имеют возможность отследить эти изменения. Если один поток открывает файл с правом чтения, другие потоки данного процесса могут читать из этого файла.

Использование потоков имеет следующие преимущества с точки зрения производительности [3]:

- Создание нового потока в уже существующем процессе занимает существенно меньше времени, чем создание нового процесса.
- Поток можно завершить быстрее, чем процесс.
- Переключение потоков в рамках одного процесса происходит намного быстрее.
- Использование потоков повышает эффективность обмена информацией между процессами. В большинстве операционных систем обмен информацией между процессами происходит с участием ядра, обеспечивающего механизм осуществления обмена и защиту. Обмен информацией между потоками может производиться без участия ядра.

2.4.1. Многопоточная модель процесса

Однопоточная модель процесса включает управляющий блок процесса, пользовательское адресное пространство (программы и данные), а также стеки ядра и пользователя, с помощью которых осуществляются вызовы процедур и возвраты из них при выполнении процесса. Многопоточная модель процесса, помимо управляющего блока процесса и адресного пространства пользователя, включает до-

полнительно для каждого потока счетчик команд, регистры, стек, локальную память, состояние выполнения.

- Счетчик команд – отслеживает порядок выполнения.
- Регистры – используются для хранения текущих переменных.
- Стек – содержит протокол выполнения процесса, где для каждой вызванной, но еще не вернувшейся процедуры, отведен отдельный фрейм. Во фрейме находятся локальные переменные процедуры и адрес возврата. Каждый поток может вызывать различные процедуры и соответственно иметь различный протокол выполнения процесса.
- Локальная память – статическая память, выделяемая потоку для локальных переменных.
- Состояние выполнения потока – одно из четырех основных состояний (готовый к выполнению, выполняющийся, блокированный, завершающийся).

В многопоточном режиме процессы обычно запускаются с одним потоком. Этот поток может создавать новые потоки, определив их указатели команд и аргументы. Новые потоки создаются со своим собственным контекстом регистров и стековым пространством, после чего помещаются в очередь готовых к выполнению потоков. В случае необходимости ожидания некоторого события поток блокируется (при этом сохраняется содержимое его пользовательских регистров, счетчика команд и указателя стека). После этого процессор может перейти к выполнению другого потока. После завершения потока его контекст регистров и стеки удаляются.

Многопоточная модель процесса охватывает две категории потоков: потоки на уровне пользователя (user-level threads – ULT) и потоки на уровне ядра (kernel – level threads – KLT) [3,5]. Потоки на уровне пользователя управляются самим приложением. Обычно приложение в начале своей работы состоит из одного потока, с которого

начинается выполнение данного приложения, и который размещается в процессе, управляемом ядром. Приложение может создать новый поток при помощи вызова библиотечной процедуры работы с потоками, например thread_create(). В результате создается структура данных для нового потока и управление передается к одному из готовых к выполнению потоков данного процесса в соответствии с заданным алгоритмом планирования. При этом контекст текущего потока сохраняется. При возврате управления к данному потоку его контекст восстанавливается. Все управление ULT осуществляется в пользовательском пространстве в рамках одного процесса. Связь с ядром отсутствует. Ядро продолжает осуществлять планирование процесса как единого целого. Однако существует взаимосвязь между планированием потоков и планированием процессов. В случае возникновения прерывания процесса, например по вводу-выводу или таймеру, вы-

полняющийся поток процесса продолжает оставаться в состоянии выполнения, хотя перестает выполняться на процессоре. При возврате управления процессу возобновляется выполнение потока на процессоре.

Использование потоков на уровне пользователя обладает определенными преимуществами по сравнению с использованием потоков на уровне ядра [3]. К числу таких преимуществ относятся:

- Для управления потоками процессу не нужно переключаться в режим ядра, что позволяет избежать дополнительных расходов.
- Планирование потоков может производиться в зависимости от специфики приложения. Для одних приложений лучше подходит простой алгоритм круговой диспетчеризации, а для других - алгоритм планирования с использованием приоритетов.
- Использование потоков на уровне пользователя применимо для любой операционной системы. Библиотека потоков представля-

ет собой набор процедур, работающих на уровне приложения и совместно используемых всеми приложениями.

К числу недостатков использования ULT относятся:

- При выполнении операционной системой системного вызова блокируется не только данный поток, но и все другие потоки данного процесса.
- Поскольку ядро закрепляет за каждым процессом только один процессор, несколько потоков одного процесса не могут выполняться одновременно даже в случае многопроцессорной системы.

Потоки на уровне ядра управляются самим ядром через интерфейс прикладного программирования (API) средств ядра, управляющих потоками [3]. Ядро поддерживает контекст процесса, а также контексты каждого отдельного потока процесса. Планирование выполняется ядром исходя из состояния потоков. Благодаря этому ядро может осуществлять планирование работы нескольких потоков одного процесса на нескольких процессорах. Кроме того, при блокировке одного из потоков процесса, ядро может выбрать для выполнения другой поток данного процесса. Основным недостатком использования потоков на уровне ядра являются дополнительные накладные расходы на переключения между потоками внутри одного процесса за счет переходов в режим ядра.

Если для большинства потоков приложения необходим доступ к ядру, то использование схемы KLT более целесообразно. Примерами использования потоков на уровне ядра являются ОС W2K, Linux. В некоторых операционных системах, например в ОС Solaris, используется комбинированный подход [3]. Потоки на пользовательском уровне, входящие в приложение, отображаются в такое же или меньшее число потоков на уровне ядра (рис 2.5). При этом число потоков на уров-

37

не ядра является изменяемым параметром, что позволяет оптимизировать работу приложения.

Рис 2.5. Потоки на уровне пользователя и ядра

2.4.2. Примеры реализации потоков

Рассмотрим простые примеры многопоточного программирования на языке C в среде ОС Linux [9]. В Linux реализована библиотека API-функций работы с потоками, которая называется Pthreads. Все функции и типы данных библиотеки объявлены в файле <pthread.h>. Эти функции не входят в стандартную библиотеку

38

языка C, поэтому при компоновке программы необходимо задавать опцию -lpthread в командной строке.

Для создания потока используется функция pthread_create(). Данной функции передаются следующие параметры:

- Указатель на переменную типа pthread_t, в которой сохраняется идентификатор нового потока. При ссылке на идентификаторы потоков в программах на C или C++ необходимо использовать тип данных pthread_t.
- Указатель на объект атрибутов потока. Этот объект определяет взаимодействие потока с остальной частью программы. Если задать его равным NULL, поток будет создан со стандартными атрибутами.
- Указатель на потоковую функцию. Функция имеет следующий тип: void* (*) (void*)
- Значение аргумента потока (тип void*). Данное значение передается потоковой функции.

При вызове программы Linux создает для нее новый процесс с единственным потоком, последовательно выполняющим программный код. Этот поток может создавать дополнительные потоки, ко-

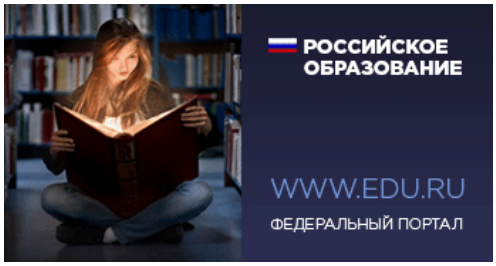
торые находятся в одном процессе и выполняют части программного кода. Ниже приводится программа создания нового потока, который совместно с главным потоком изменяет разделяемую статическую глобальную переменную. Для вывода идентификатора выполняемого в текущий момент потока используется функция pthread_self(). Для того, чтобы родительский поток не завершился раньше дочернего, используется функция pthread_join(). Данная функция принимает два аргумента: идентификатор ожидаемого потока и указатель на переменную

void*, в которую будет записано значение, возвращаемое потоком. Если данное значение не важно, в качестве второго аргумента задается NULL.

```
    /* thread1.c

    * This program creates one new thread.
    * New thread and main thread increase static shared variable on 1.
    * Each thread in a process is identified by a thread ID.

     to compile: gcc -o thread1 thread1.c -lpthread
    http://www.intuit.ru/department/os/osintropractice/examples.html     */

 #include <pthread.h>
 #include <stdio.h>


 int a = 0;

 void *mythread(void *dummy)
 {
     pthread_t mythid;
     mythid = pthread_self();
     a = a+1;
     printf("Thread-1 %d, Calculation result = %d\n",
 mythid, a);
     return NULL;
 }

 int main()
 {
     pthread_t thid, mythid;
     int result;
```

Страницы ← предыдущая следующая →

1 2 3 4 5 6 7 8 9 10



Минпросвещения России

ФГАУ ГНИИ ИТТ «Информика»

Рособрнадзор

Федеральная университетская компьютерная сеть РФ

Федеральный портал "Российское образование"

Единое окно доступа к образовательным ресурсам

© 2005-2019 ФГАУ ГНИИ ИТТ "Информика"

Главная | Каталог | Избранное | Порталы | Библиотеки ВУЗов | Отзывы | Новости | Разработчикам сайтов | Рекламодателям | Контакты | Карта сайта