электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.9

Обзор библиотек для многопоточного программирования на языке C

***Сорокин Д. А.**, студент*
Россия, 105005, г. Москва, МГТУ им. Н. Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

Научный руководитель: Рудаков И. В., к.т.н., доцент
кафедра «Программное обеспечение ЭВМ и информационные технологии»,
Россия, 105005, г. Москва, МГТУ им. Н. Э. Баумана
irudakov@bmstu.ru

Введение

В наши дни стали широко распространены компьютеры с многоядерными процессорами, и с их развитием появилась востребованность в параллельных программах. Однако следует отчётливо разделять два класса параллелизма (и требуемых для их обеспечения синхронизаций), природа которых различного происхождения [1]:

- Логический параллелизм (или квази-параллелизм), при котором потоки вытесняют друг друга, создавая иллюзию параллельности. При этом синхронизация осуществляется классическими блокирующими механизмами, когда поток ожидает недоступных ему ресурсов, переводясь в блокированное состояние.
- Физический параллелизм (или реальный параллелизм), возникший с широким распространением архитектуры SMP (*англ. Symmetric Multiprocessing – симметричное мультипроцессирование*), когда разные задачи ядра выполняются одновременно на различных процессорах. В этом случае используются примитивы синхронизации, когда один из процессоров ожидает требуемых ресурсов.

Параллельность выполнения программы может быть обеспечена несколькими путями. В данной работе рассматривается параллельное выполнение потоков. Поток выполнения (*тред; от англ. thread – нить*) – наименьшая единица обработки, исполнение которой может быть назначено ядром операционной системы. Среди достоинств такого подхода можно выделить следующие:

- Возможность повышения производительности программы за счет распараллеливания процессорных вычислений и операций ввода-вывода.
- Для создания потока требуется меньше времени, чем для создания процесса.

- В некоторых случаях использование потоков позволяет упростить архитектуру программы за счёт общего адресного пространства между потоками.

Для создания параллельных многопоточных программ на языке C существует несколько библиотек, из которых наиболее распространёнными являются следующие три: *Pthreads*, *win32 threads* и *glib threads*. Но перед тем, как рассмотреть эти программные библиотеки, рассмотрим примитивы синхронизации, доступные них.

Примитивы синхронизации

Набор примитивов синхронизации, доступных параллельной многопоточной программе, зависит как от выбора целевой платформы, на которой будет развернута программа, так и от выбора библиотеки, которая предоставляет программный интерфейс (API) для работы с этими примитивами. Ниже перечислены те примитивы, которые представлены на большинстве платформ и у большинства программных библиотек.

Семафор (*англ. semaphore*) – примитив синхронизации, используемый для управления доступом к ресурсам среди нескольких потоков (или процессов). Семафор можно представить как переменную, хранящую число свободных ресурсов. Его значение не может опускаться ниже нуля.

Над семафором можно применять две операции – V (сигнал) и P (ожидание). Операция V увеличивает значение семафора на единицу, а операция P – уменьшает на единицу. В случае, если поток пытается захватить семафор (применить операцию P) при значении семафора, равном нулю, то выполнение данного потока будет приостановлено до тех пор, пока любой другой поток не выполнит операцию V.

Мьютекс (*англ. mutex*) – частный случай семафора, когда число его ресурсов равно единице. То есть, мьютекс может находиться только в двух состояниях – заблокированном или свободном. В случае, если поток попытается захватить уже занятый мьютекс, то этот поток «заснёт» до тех пор, пока другой поток не разблокирует этот мьютекс.

Мьютекс чтения-записи (*англ. read/write mutex*) – мьютекс, который решает проблему читателей/писателей. Такой мьютекс позволяет захват двух типов: захват со стороны читателя и захват со стороны писателя. При этом такой мьютекс может быть захвачен сразу несколькими читателями, но не может быть захвачен несколькими писателями. Кроме того, пока мьютекс захвачен хотя бы одним читателем, никакой писатель но может захватить этот мьютекс, и наоборот (никакой читатель не может захватить мьютекс, пока его удерживает писатель).

Спинлок (*англ. spin-lock*) – примитив синхронизации, схожий с мьютексом, но при ожидании которого поток не блокируется, а продолжает своё выполнение, выполняя холостые циклы ожидания. Спинлоки применяются для синхронизации выполнения небольших участков кода, время выполнения которых меньше времени переключения контекста.

События (*англ. event*) – примитив синхронизации, используемый для уведомления ожидающих потоков о том, что произошло какое-либо событие. Событие позволяет применять следующие операции:

1. *wait* – при вызове приостанавливает выполняющийся поток до тех пор, пока не произойдёт событие;
2. *set* – помечает событие как свершившееся; все ожидающие потоки на этом событии «просыпаются» и продолжают своё выполнение;
3. *clear* – помечает событие как непроизошедшее.

При этом событие, как правило, запоминает своё состояние, и если событие произошло, то все потоки, которые вызовут операцию *wait*, не будут приостановлены и продолжат своё выполнение.

Условные переменные (*англ. conditional variables*) – примитивы синхронизации, обеспечивающие блокирование одного или нескольких потоков выполнения до момента наступления сигнала от другого потока о выполнении некоторого условия. По своему назначению условные переменные схожи с событиями, но имеют свои особенности. Условные переменные, как правило, используются совместно с ассоциированным мьютексом. Условная переменная не хранит своё состояние, и поэтому в случае, если условие выполнится раньше, чем поток остановится на данной переменной, то произошедшее условие будет потеряно.

Барьер (*англ. barrier*) – примитив синхронизации, который приостанавливает выполнение группы потоков до тех пор, пока все потоки из этой группы не достигнут определённой точки выполнения в программе.

POSIX Threads

POSIX (*англ. Portable Operating System Interface for Unix — переносимый интерфейс операционных систем Unix*) — набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой (системный API), библиотеку языка C и набор приложений и их интерфейсов. Стандарт создан для обеспечения совместимости различных UNIX-подобных операционных систем и переносимости

прикладных программ на уровне исходного кода, но может быть использован и для не-Unix систем.

Библиотека *Pthreads* предоставляет доступ к следующим примитивам синхронизации: мьютексам, мьютексам чтения/записи, спинлокам, условным переменным и к барьерам. Стандарт *POSIX* также предоставляет доступ к семафорам, но они не включены в состав библиотеки *Pthreads*.

Все функции библиотеки для работы с потоками и примитивами синхронизации начинаются с префикса «*pthread_**». Для управления циклом жизни потока используются следующие функции: *pthread_create* для создания, *pthread_cancel* для завершения потока извне, *pthread_exit* для корректного завершения потока изнутри, *pthread_join* для ожидания завершения потока и т. п. При создании потока библиотека позволяет задать некоторые его параметры, например, размер стека, алгоритм планирования, а также будет ли он потоком уровня ядра или потоком уровня пользователя [3].

Для работы с примитивами синхронизации используются функции *pthread_mutex**, *pthread_cond**, *pthread_rwlock**, *pthread_spin**, *pthread_barrier** и т. п.

Win32 Threads

Системный API (*англ. Application Programming Interface – интерфейс прикладного программирования*) операционной системы Windows (*win32 API*) предоставляет свой собственный набор функций и типов данных, позволяющих разрабаотывать многопоточные приложения.

Win32 API предоставляет доступ к следующим примитивам синхронизации: семафорам, мьютексам, спинлокам и событиям. Для хранения этих примитивов используется единый тип – *HANDLE*, а для попытки захвата объекта используется одна функция – *WaitForSingleObject(…)*. Для захвата сразу нескольких примитивов синхронизации используется функция *WaitForMultipleObjects(…)*, при этом захват всех примитивов происходит атомарно [6].

Для работы с потоками используются следующие функции: *CreateThread* и *ExitThread*. Для создания примитивов синхронизации применяют функции с префиксом *Create**, а для закрытия – *CloseHandle*. Таким образом, в *win32 API* используется гораздо меньше функций, чем в библиотеке *pthreads*, но их применение может быть не так очевидно и прозрачно, как в *pthreads*.

Следует отметить, что существует несколько неофициальных проектов, которые реализуют интерфейс потоков *POSIX* для операционной системы *Windows*. Среди них

следует отметить *pthreads-win32* [4] и *pthreads4w* [5], однако они не полностью реализуют функционал, доступный на *UNIX*-подобных платформах.

Glib Threads

Glib – низкоуровневая библиотека, расширяющая возможности стандартной библиотеки языка C *libc*. Данная библиотека разрабатывается в рамках и лежит в основе проектов *GTK+* и *Gnome*. Изначально она разрабатывалась для *UNIX*-подобных операционных систем, но позже была портирована и для операционных систем *Windows* и *Mac OS*.

Glib предоставляет следующие объекты для работы в многопоточной среде – *GThread* (поток), *GMutex* (мьютекс), *GRecMutex* (рекурсивный мьютекс, который подсчитывает число захватов и возвратов), *GRWLock* (мьютекс чтения/записи), *GCond* (условная переменная), а также некоторые другие средства, например, *GOnce* (позволяет выполнить участок кода единожды в пределах одного потока). Функции для работы с этими объектами имеют соответствующие префиксы, например, *g_mutex_**, *g_cond_**, *g_rw_lock_** и т. п.

Рассмотрев программные библиотеки для разработки параллельных многопоточных приложений, приведём сравнение доступных примитивов синхронизации у рассмотренных библиотек (таблица ниже).

Примитивы	**Pthreads**	**Win32 Threads**	**Glib Threads**
Семафор		+	
Мьютекс	+	+	+
Мьютекс чтения/записи	+		+
Спинлок	+	+	
Событие		+	
Условная переменная	+		+

Примитивы	Pthreads	Win32 Threads	Glib Threads
Барьер	+		

Заключение

В данной работе были рассмотрены особенности разработки параллельных многопоточных программ, их достоинства и недостатки по сравнению с традиционными программами с последовательным выполнением команд. Были рассмотрены основные примитивы синхронизации выполнения потоков, выделены их особенности и области применения. Выделены наиболее распространённые программные библиотеки для разработки параллельного многопоточного приложения, описаны предоставляемые ими программный интерфейс и примитивы синхронизации.

Список литературы

1. Цилюрик О. И. Модули ядра Linux. 2011. 218 с. Режим доступа: http://rus-linux.net/MyLDP/BOOKS/Moduli-yadra-Linux/KERN-modul-4.95.pdf (дата обращения 26.03.15).
2. Цилюрик О. И., Горошко Е. QNX/UNIX: анатомия параллелизма. СПб.: «Символ-Плюс», 2005. 288 с. ISBN 5-93286-088-X.
3. Blaise B. POSIX Threads Programming // Lawrence Livermore National Laboratory. 2014. Режим доступа: https://computing.llnl.gov/tutorials/pthreads/ (дата обращения 25.03.15).
4. Open Source POSIX Threads for Win32. 2012. Режим доступа: https://www.sourceware.org/pthreads-win32/ (дата обращения 25.03.15).
5. An implementation of the POSIX threads API for Windows. 2012. Режим доступа: http://sourceforge.net/projects/pthreads4w/ (дата обращения 25.03.15).
6. Sayegh R. Comparison between threading mechanism in Win32 and POSIX systems. 2011. Режим доступа: http://www.slideshare.net/abufayez/pthreads-vs-win32-threads (дата обращения 25.03.15).
7. Сорокин Д. А. Использование сетей Петри для поиска тупиков в вычислительных системах с общей памятью // Молодежный научно-технический вестник. МГТУ им. Н.Э. Баумана. Электрон. журн. 2014. № 12. Режим доступа: http://sntbul.bmstu.ru/doc/745958.html (дата обращения 31.03.2015).

8. Пащенкова А. В. Метод формализации программного обеспечения иерархическими сетями Петри // Молодежный научно-технический вестник. МГТУ им. Н.Э. Баумана. Электрон. журн. 2013. № 6. Режим доступа: http://sntbul.bmstu.ru/doc/577675.html (дата обращения 31.03.2015).
9. Glib threads. 2014. Режим доступа: https://developer.gnome.org/glib/stable/glib-Threads.html (дата обращения 25.03.15).