ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

УДК 681.3

В. В. Никифоров, В. И. Шкиртиль

СОСТАВНОЕ БЛОКИРОВАНИЕ ВЗАИМОСВЯЗАННЫХ ЗАДАЧ В СИСТЕМАХ НА МНОГОЯДЕРНЫХ ПРОЦЕССОРАХ

Проводится сравнение свойств протоколов доступа к разделяемым ресурсам в многозадачных программных приложениях при их исполнении с использованием классических (одноядерных) и многоядерных процессоров. Показано, что некоторые свойства протоколов, полезные при реализации многозадачных приложений средствами одноядерных процессоров, становятся недействительными при реализации средствами многоядерных процессоров.

Ключевые слова: *многозадачные системы, системы на многоядерных процессорах, системы реального времени, взаимосвязанные задачи, протоколы доступа к ресурсам.*

Введение. Программные приложения, разрабатываемые для использования в различных сферах применения компьютерной техники, строятся преимущественно в виде комплексов *задач* — замкнутых по управлению последовательных программ, служащих достижению общих стоящих перед системой целей и разделяющих общие системные ресурсы [1]. В системах имитационного моделирования каждый отдельный процесс требует представления в виде отдельной задачи, что предопределяет необходимость построения системы в виде многозадачного комплекса. Во встроенных компьютерных системах в системах реального времени требование построения программного приложения в виде комплекса кооперативных задач диктуется принципом соответствия структуры совокупности внешних процессов, контролируемых системой, структуре программного комплекса [2]. В связи с распространением аппаратных средств на многоядерных процессорах для обеспечения эффективной загрузки процессорных ядер программные приложения информационно-вычислительного характера целесообразно строить также в виде многозадачных программных систем.

Составляющие программное приложение задачи τ_1, τ_2, …, τ_n разделяют между собой имеющиеся в системе *исполнительные* ресурсы (процессоры, ядра многоядерного процессора) соответственно принятой дисциплине планирования, выражаемой, например, статическим назначением *приоритетов* задач — целочисленных параметров, определяющих порядок исполнения задач [3, 4].

Наряду с исполнительными ресурсами задачи могут разделять имеющиеся общесистемные *информационные* ресурсы (глобальные массивы, буферы, интерфейсные регистры, экранные формы). Для обеспечения целостности информационных ресурсов, разделяемых взаимосвязанными задачами, на границах участков кода задачи τ_i, в рамках которых исполняются действия с разделяемым информационным ресурсом, предусматриваются операции над *мьютексами* [1]. Такие участки кода называются критическими интервалами по доступу

задачи τ_i к разделяемому ресурсу. Конкретный мьютекс является синхронизирующим элементом, предотвращающим одновременный доступ различных задач к конкретному ресурсу. При запросе высокоприоритетной задачей τ_i доступа в очередной критический интервал она может быть приостановлена до момента, когда некая низкоприоритетная задача τ_j выйдет из своего критического интервала по доступу к этому ресурсу. Такая ситуация называется *блокированием* (низкоприоритетная задача τ_j блокирует высокоприоритетную задачу τ_i). *Составное блокирование* означает, что в рамках цикла исполнения задача неоднократно блокируется низкоприоритетными задачами [5].

Задачи, выполнение которых не может быть приостановлено синхронизирующими механизмами (например, выполнением операций над мьютексами), называются *независимыми*. Независимым задачам противопоставляются *взаимосвязанные* задачи, которые могут попадать в состояния ожидания тех или иных действий, выполняемых другими задачами (например, освобождения занятых общесистемных информационных ресурсов). При построении систем с взаимосвязанными задачами необходимо гарантировать невозможность возникновения *взаимных ожиданий* — ситуаций, в которых между действующими задачами возникают кольца отношений блокирования [6].

Исследования, направленные на поиск эффективных методов построения многозадачных приложений на классических одноядерных процессорах, выполняются с 1970-х гг. В результате этих исследований разработаны, в частности, оптимальные дисциплины планирования, протоколы доступа к разделяемым информационным ресурсам, гарантирующие отсутствие опасности возникновения взаимных ожиданий, методы оценки времени отклика задач [7]. Аналогичные исследования для систем на многоядерных процессорах особенно интенсивно проводятся в последнее десятилетие. Так, установлено, что ряд тезисов, справедливых для систем, состоящих из независимых задач, на классических процессорах, становится недействителен для систем на многоядерных процессорах. Например, дисциплины планирования, оптимальные для классических систем, не являются оптимальными для многоядерных [8—10]. Наихудший (в отношении своевременности выполнения задач) сценарий событий для классических систем не является наихудшим для многоядерных систем [11]. В настоящей статье показано, что протоколы доступа к ресурсам, гарантирующие для одноядерных процессоров отсутствие опасности составного блокирования и взаимных ожиданий задач, при реализации на многоядерных процессорах могут терять эти полезные свойства.

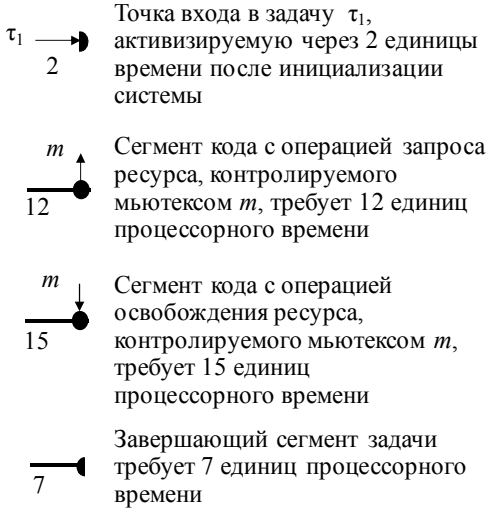

Рис. 1

Протоколы доступа к критическим интервалам. Изучение свойств многозадачных приложений базируется на построении моделей. К таким моделям относятся маршрутные сети [6], в которых составляющие приложение задачи представляются последовательностями сегментов, разделяемых операциями над мьютексами. Типы элементов маршрутных сетей, используемых в рамках настоящей статьи, приведены на рис. 1.

На рис. 2, *а* представлена конфигурация некоторого программного приложения A_1 из трех задач (задачи индексируются в порядке снижения приоритета) с использованием маршрутных сетей. Порядок следования системных событий и смены состояний составляющих приложение объектов в ходе работы системы отображается диаграммами, которые иллюстрируют ход исполнения приложения A_1 при использовании простейшего протокола доступа к ресурсам (рис. 2, *б*) и протокола наследования

приоритетов (рис. 2, *в*). Протокол устанавливает специфику исполнения операции запроса ресурса, а именно: требуемые условия для выдачи задаче разрешения занять запрашиваемый ресурс и побочные системные эффекты, возникающие при входе в критический интервал.

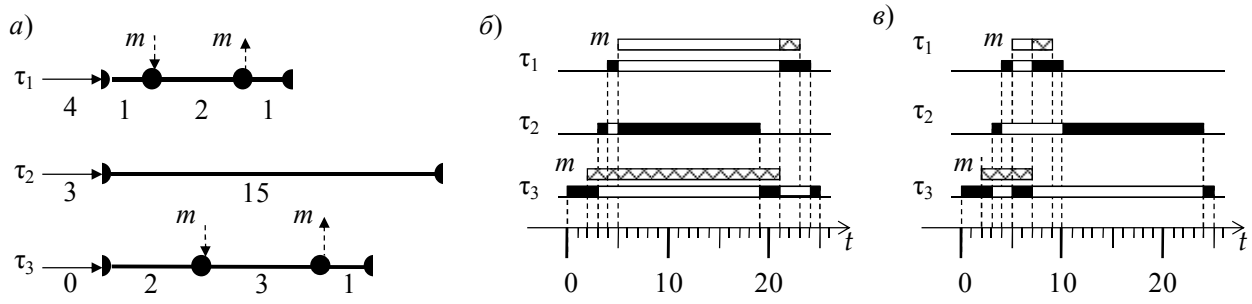

Рис. 2

Простейший протокол доступа к ресурсам и протокол наследования приоритетов. При использовании простейшего протокола (ПП) условия предоставления запрашиваемого ресурса ограничиваются единственным (обязательным для всех протоколов) требованием: ресурс должен быть свободен. Никаких дополнительных побочных эффектов (кроме обязательного — ресурс переводится в состояние „занят“) при входе в критический интервал не возникает. В таблице описаны события, возникающие на начальном участке исполнения приложения A_1 при использовании ПП (см. рис. 2, *б*); в таблице ресурс обозначен тем же символом *m*, что и охраняющий его мьютекс.

Момент времени	Событие
t=0	Активизация задачи τ_3, получение ресурса процессора и начало исполнения
t=2	Запрос задачей τ_3 ресурса *m*, захват ресурса и продолжение исполнения
t=3	Активизация задачи τ_2, вытеснение задачи τ_3 и начало исполнения
t=4	Активизация задачи τ_1, вытеснение задачи τ_2 и начало исполнения
t=5	Начало ожидания задачей τ_1 освобождения ресурса *m*, возобновление исполнения задачи τ_2
t=19	Завершение исполнения задачи τ_2, возобновление исполнения задачи τ_3
t=21	Освобождение задачей τ_3 ресурса *m*, захват задачей τ_1 ресурса *m* и вытеснение задачи τ_3
t=23	Завершение исполнения задачи τ_3

На интервале $t \in (5, 21)$ высокоприоритетная задача τ_1 ожидает предоставления занятого ресурса *m*. Причем большую часть этого интервала процессором владеет менее приоритетная задача τ_2, не имеющая отношения к ожидаемому высокоприоритетной задачей τ_1 ресурсу *m*. Это явление называется инверсией приоритетов.

Исключить вероятность возникновения инверсии приоритетов можно путем использования протокола наследования приоритетов (ПНП). Условия предоставления запрашиваемого ресурса в случае ПНП те же, что и в случае ПП (ресурс должен быть свободен), но при запросе задачей τ_i ресурса, занятого задачей τ_j, реализуется побочный системный эффект: приоритет задачи τ_j, владеющей занятым ресурсом, временно, до ее выхода из критического интервала, повышается до приоритета задачи τ_i (τ_j наследует приоритет τ_i). Для конфигурации

приложения A_1 (см. рис. 2, *а*) использование ПНП существенно сокращает продолжительность ожидания задачей τ_i требуемого ей ресурса (см. рис. 2, *в*).

Использование ПНП не исключает возможности возникновения составного блокирования высокоприоритетных задач. Рис. 3, *б* иллюстрирует такую возможность. При исполнении приложения A_2, конфигурация которого представлена на рис. 3, *а*, к моменту активизации задачи τ_1 задачами τ_2 и τ_3 уже захвачены соответственно ресурсы m_1 и m_2. Поэтому задача τ_1 ожидает сначала освобождения ресурса m_1 на интервале $t\in(5, 14)$, затем — освобождения ресурса m_2 на интервале $t\in(18, 27)$. Как будет показано далее, использование ПНП не исключает возможности возникновения взаимного блокирования.

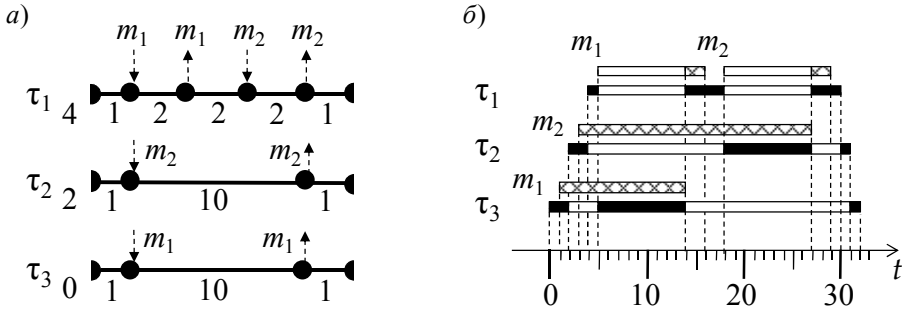

Рис. 3

Протоколы доступа к ресурсам на основе пороговых приоритетов. Исключить вероятность возникновения взаимного и составного блокирования задач можно с помощью протоколов доступа к ресурсам, использующих *пороговые приоритеты* ресурсов — статически определяемые параметры каждого из разделяемых ресурсов: пороговый приоритет ресурса равен высшему уровню статического приоритета задачи, которая может занять ресурс. Известны две разновидности протоколов такого типа — протокол пороговых приоритетов (ППП) и протокол превентивного наследования приоритетов (ППНП) [4].

ППП характеризуется тем, что в дополнение к механизму ПНП ставится условие предоставления запрашиваемого ресурса: запрашиваемый ресурс предоставляется задаче только тогда, когда ее статический приоритет выше, чем максимальный из пороговых приоритетов всех ресурсов, уже занятых в текущий момент времени другими задачами.

ППНП характеризуется усилением (относительно ПНП) побочного системного эффекта, возникающего при входе в критический интервал: при предоставлении задаче τ_i затребованного ресурса ее приоритет немедленно повышается до значения порогового приоритета захватываемого ресурса; такое повышенное значение приоритета задачи τ_i сохраняется до освобождения ею предоставленного ресурса.

Известно, что при исполнении приложений с взаимодействующими задачами на одноядерных процессорах как ППП, так и ППНП гарантируют невозможность возникновения составного блокирования. На рис. 4, *а*, *б* соответственно представлены диаграммы, иллюстрирующие предотвращение составного блокирования при использовании ППП и ППНП в ходе исполнения приложения A_2 (см. рис. 3, *а*).

Пороговые приоритеты разделяемых ресурсов m_1 и m_2 равны приоритету задачи τ_1. При использовании ППП (см. рис. 4, *а*) в момент t=3 задача τ_2 запрашивает свободный ресурс m_2. Несмотря на то что ресурс m_2 свободен, его предоставление задаче τ_2 откладывается, поскольку занят ресурс m_1, обладающий наивысшим пороговым приоритетом. В момент t=5 задача τ_1 запрашивает ресурс m_1 и переходит в состояние ожидания момента его освобождения (до t=13). При использовании ППНП (см. рис. 4, *б*) в момент t=1 задача τ_3 получает дос-

туп к ресурсу m_1, имеющему наивысшее значение порогового приоритета; на время владения этим ресурсом приоритет задачи τ_3 становится наивысшим.

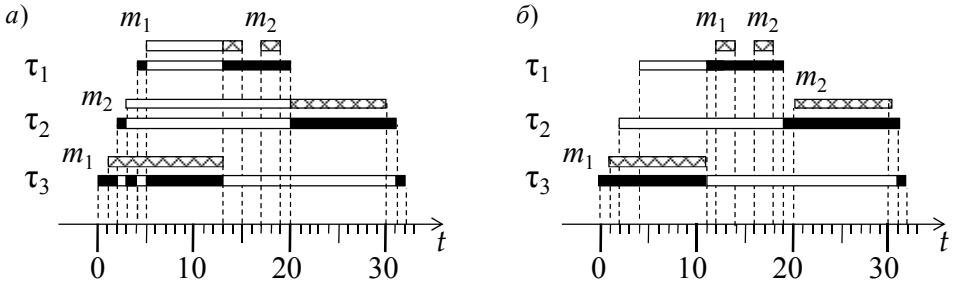

Рис. 4

На рис. 5, *а* приведена конфигурация приложения A_3, которое при использовании ПНП (рис. 5, *б*) попадает в состояние взаимного блокирования задач. Анализ показывает, что при использовании ППП (рис. 5, *в*) или ППНП (рис. 5, *г*) возможность возникновения взаимного блокирования исключается. В системах на одноядерных процессорах использование ППП и ППНП гарантирует невозможность взаимного блокирования задач при любых конфигурациях приложений.

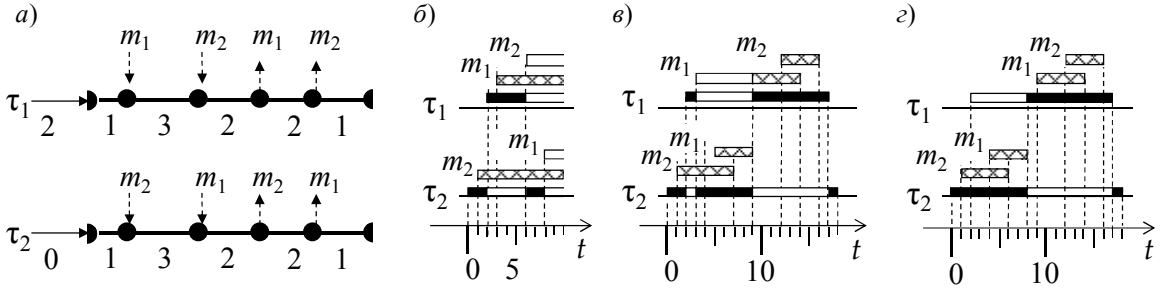

Рис. 5

Составное и взаимное блокирование в системах на многоядерных процессорах. Некоторые из отмеченных выше особенностей исполнения одноядерными процессорами программных приложений различных конфигураций с использованием определенных протоколов доступа к ресурсам не соблюдаются при исполнении на многоядерных процессорах.

Диаграмма, приведенная на рис. 6, *а*, показывает, что, в отличие от исполнения одноядерным процессором с ПНП приложения A_2 (см. рис. 3, *а*), при исполнении двуядерным процессором того же приложения с тем же протоколом составное блокирование задачи τ_1 не возникает. Можно привести такой сценарий системных событий, при котором составное блокирование на двуядерном процессоре возникает и при использовании ППНП (что исключено в случае одноядерного процессора). Отметим также, что время отклика задачи τ_1 при использовании ППП (рис. 6, *б*) превышает время ее отклика при использовании ПНП (см. рис. 6, *а*). При исполнении на одноядерном процессоре такого превышения нет.

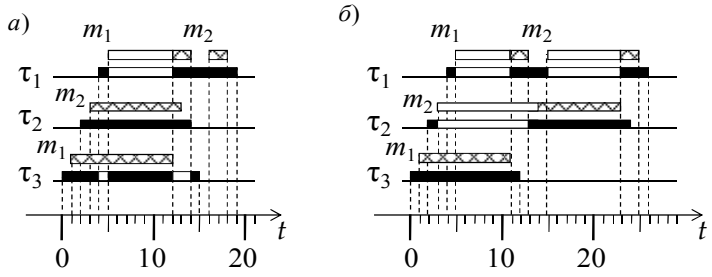

Рис. 6

Конфигурация программного приложения A_3 (см. рис. 5, *а*) при исполнении на одноядерном процессоре с использованием ПП или ПНП приводит к взаимному блокированию

задач. При этом использование ППП и ППНП избавляет многозадачные системы на одноядерных процессорах от возникновения такого рода некорректных ситуаций.

Диаграмма, приведенная на рис. 7, *а*, показывает, что в отличие от систем на одноядерных процессорах при использовании ППНП на многоядерном процессоре возможно возникновение взаимного блокирования задач. Исполнение того же приложения на том же двуядерном процессоре с использованием ППП позволяет избежать возникновения ситуации взаимного блокирования (рис. 7, *б*).

Таким образом, как показывает анализ, использование ППП гарантирует невозможность взаимного блокирования не только для систем на одноядерных процессорах, но и для систем на многоядерных процессорах.

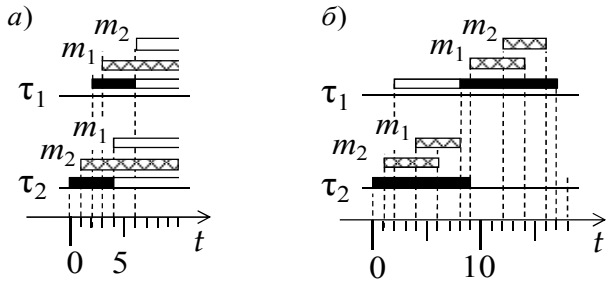

Рис. 7

Заключение. Ряд закономерностей исполнения классическими одноядерными процессорами программных приложений различных конфигураций с использованием определенных протоколов доступа к ресурсам недействителен при исполнении приложений многоядерными процессорами. Так, в системах на классических процессорах применение как ППП, так и ППНП предотвращает взаимное блокирование задач. В системах на многоядерных процессорах ППП сохраняет это полезное свойство, но ППНП его теряет. ППП и ППНП, предотвращающие составное блокирование на одноядерных процессорах, не предотвращают его в случае построения системы на многоядерном процессоре. При использовании ППП на одноядерном процессоре время отклика высокоприоритетной задачи не превосходит времени ее отклика при использовании ПНП. Для многоядерного процессора такая закономерность нарушается. Возможность возникновения составного блокирования задач приводит к увеличению значений фактора блокирования, что необходимо учитывать при оценке гарантий своевременности выполнения задач в системах реального времени.

СПИСОК ЛИТЕРАТУРЫ

1. *Никифоров В. В.* Операционные системы реального времени. СПб: Изд-во Рос. гос. пед. ун-та, 2007. 109 с.

2. *Давиденко К. Я.* Технология программирования АСУТП. Проектирование систем реального времени, параллельных и распределенных приложений. М.: Энергоатомиздат, 1985. 183 с.

3. *Никифоров В. В., Павлов В. А.* Операционные системы реального времени для встроенных программных комплексов //Программные продукты и системы. 1999. № 4. С. 24—30.

4. *Данилов М. В.* Методы планирования выполнения задач в системах реального времени // Там же. 2001. № 4. С. 28—35.

5. *Никифоров В. В., Шкиртиль В. И.* Управление задачами в системах реального времени // Материалы Первой Междунар. конф. „Автоматизация управления и интеллектуальные системы и среды“. Нальчик: Изд-во КБНЦ РАН, 2010. Т. II. С. 139—143.

6. *Никифоров В. В., Шкиртиль В. И.* Маршрутные сети — графический формализм представления структуры программных приложений реального времени // Тр. СПИИРАН / Под общ. ред. *Р. М. Юсупова*. 2010. Вып. 14. С. 7—28.

7. *Liu J.W.S.* Real-Time Systems. NJ: Prentice Hall, 2000.

8. *Dhall S. K., Liu C. L.* On a real-time scheduling problem // Operating Research. 1978. Vol. 26, N 1. P. 127—140.

9. *Baker T.* Multiprocessors EDF and deadline monotonic schedulability analysis // Proc. of the 24th IEEE Real-Time Systems Symposium. 2003. P. 120—129.

10. *Никифоров В. В.* Выполнимость приложений реального времени на многоядерных процессорах // Тр. СПИИРАН / Под общ. ред. *Р. М. Юсупова*. 2009. Вып. 8. С. 255—284.

11. *Никифоров В. В., Шкиртиль В. И.* Использование многоядерных процессоров для построения систем реального времени // Изв. вузов. Приборостроение. 2007. Т. 50, № 10. С. 28—35.

Сведения об авторах

Виктор Викентьевич Никифоров — д-р техн. наук, профессор; Санкт-Петербургский институт информатики и автоматизации РАН, лаборатория технологий и систем программирования; E-mail: nik@iias.spb.su

Вячеслав Иванович Шкиртиль — канд. техн. наук, доцент; Санкт-Петербургский институт информатики и автоматизации РАН, лаборатория технологий и систем программирования; E-mail: jvatlas@mail.rcom.ru

Рекомендована СПИИРАН

Поступила в редакцию 22.06.11 г.