ПОСТРОЕНИЕ СЛУЖБЫ СИНХРОНИЗАЦИИ ПРОЦЕССОВ ЦИКЛА РЕАЛЬНОГО ВРЕМЕНИ

Иванов А.Ю., Круглый А.А.
Кафедра ЭВМ ДонГТУ
ai@cs.dgtu.donetsk.ua

Abstract

Ivanov A.Y.,Krugluy A.A. Design serve for synchronization of process of cycle of real time. *The problems of designing of real-time software of computer are described. Parameters of Windows are researched. Multitasking by thread is used for synchronization of process. Manager is developed for control of periodical task of real-time.*

Проектирование систем реального времени и соответствующих инструментальных средств, требует использования операционных сред, обеспечивающих параметры [1] критичные для службы синхронизации процессов реального времени (РВ). Операционные системы (ОС) общего назначения, в особенности многопользовательские (например, UNIX), ориентированы на оптимальное распределение ресурсов компьютера между пользователями и задачами. В ОС РВ главная задача – успеть отреагировать на события, которые происходят на объекте управления. Система РВ, как программно-аппаратный комплекс, неотделима от датчиков регистрации событий на объекте, модулей ввода-вывода информации в цифровую часть. Системы жесткого и мягкого РВ предоставляют различные по параметрам следующие механизмы управления задачами: система приоритетов и алгоритмы диспетчеризации, механизмы межзадачного взаимодействия, средства для работы с таймерами. В многозадачных ОС общего назначения используются различные алгоритмы круговой диспетчеризации, основанные на понятии беспрерывного кванта времени, который предоставляется процессу для работы. Главный недостаток этого – на протяжении беспрерывного кванта времени, процессором владеет только одна задача. Все существующие ОС могут быть отнесены к одному из трех типов: исполнительных систем РВ, ядер РВ, ОС РВ на базе систем UNIX с алгоритмом планирования - “приоритетный с вытеснением”. Анализ публикаций [1-3, 6], свидетельствует о возможности использования Windows 2000 для решения задач РВ. Для задач с инициативным типом вызова Windows 2000 не может являться ОС жесткого РВ. Однако при периодических типах вызовов возможно ее использование, так как она позволяет [2,3]: организовывать параллельное выполнение нескольких задач в системе, имеет развитые средства обмена данными между задачами, имеет стабильную службу времени, имеет развитую подсистему приоритетного планирования процессов.

Механизм диспетчеризации ОС семейства Windows построен на базе круговой схемы с вытеснением поточного типа [3] и главным элементом механизма многозадачности является поток. Необходимо учесть, что система планирует выполнение только тех потоков, которые могут получать время процессора. Но большинство потоков к таким не принадлежит, так как большинство потоков в системе являются приостановленными. Кроме приостановленных, не планируются и другие потоки, которые ждут возникновения некоторого события. Необходимо обратить внимание также на то, что потоки с более высоким приоритетом всегда вытесняют потоки с более низким приоритетом независимого

от того, выполняются последние или нет. Если процессор выполняет поток с приоритетом 5, и в это время система обнаруживает, что поток с более высоким приоритетом готов к выполнению, то система сразу снимет поток с более низким приоритетом – даже если не завершился его квант процессорного времени.

Для Windows98 и Windows2000 исследовались следующие параметры подсистем многозадачности: время кванта работы потока, время переключения контекста потоков, время «голодания» запланированного потока. Специальная тестовая программа включала три однотипных потока, которые имели одинаковый уровень приоритета и выполняли последовательную фиксацию текущего времени. Дополнительные исследования выполнялись с помощью утилиты Microsoft Spy++ из среды разработки программных продуктов Microsoft Visual Studio 6.0., которая позволяет в Windows2000 получить следующие параметры потоков: общее время выполнения потока, время выполнения потоком кода системы, время выполнения непосредственно кода потока, количество переключений контекста потока. Во время исследования в системе функционировала программа, которая состояла из трех потоков одинакового приоритета. Для получения реальных результатов также выполнялось изменение условий системного окружения, а именно в систему прибавился еще один поток с приоритетом равным тестовым. В результате исследования была получена выборка значений исследовавшихся параметров для каждого приоритета потока и типа ОС. На основании анализа этих выборок были установленные количественные характеристики параметров. Для Windows98 получены следующие характеристики:

а) Базовый класс приоритета всех потоков Normal:
- время квантов работы потока – от 0,3 до 100 миллисекунд;
- время переключения контекста потоков – от 0,3 до 1 миллисекунды
- время «голодания» запланированного потока – от 0,3 до 600 миллисекунд.

б) Базовый класс приоритета всех потоков Real-time:
- время квантов работы потока – от 0,3 до 100 миллисекунд;
- время переключения контекста потоков – от 0,3 до 1 миллисекунды;
- время «голодания» запланированного потока – от 0,3 до 300 миллисекунд.

Для Windows2000 получены следующие характеристики:

а) Базовый класс приоритета всех потоков Normal:
- время квантов работы потока – от 10 до 25 миллисекунд;
- время переключения контекста потоков – от 0,5 до 2 миллисекунд (время переключение контекста становилось тем больше, чем большей было количество планируемых потоков в ОС);
- время «голодания» запланированного потока – от 10 до 25 миллисекунд.

б) Базовый класс приоритета всех потоков Real-time:
- время квантов работы потока – от 10 до 25 миллисекунд;
- время переключения контекста потоков – от 0,5 до 1,8 миллисекунды;
- время голодания запланированного потока – от 10 до 25 миллисекунд.

Все приведенные значения действительны для ситуации, когда в системе не присутствуют запланированные потоки с более высоким приоритетом, которые длительное время занимают процессор. На основе анализа выборок и полученных значений параметров можно сделать вывод, что Windows98 не может быть использована в качестве среды функционирование службы РВ, поскольку имеет нестабильный и не прогнозируемый алгоритм работы планировщика потоков, не может обеспечить стабильной длительности работы и заданного времени вызова потока. Для Windows2000 полученные результаты можно обобщить:

1) Планировщик потоков обеспечивает стабильную длительность кванта времени работы потока.

2) С большой достоверностью можно гарантировать вызов потока в заданное время.
3) Планировщик потоков системы имеет стабильный и прогнозируемый алгоритм управления потоками.

На основании всего выше изложенного можно сделать вывод, что Windows2000 может быть использованной в качестве среды функционирование службы РВ, так как имеет значения параметров подсистемы многозадачности на уровне систем жесткого РВ. При объединении алгоритма установки приоритетов для задач РВ и механизма планирования потоков ОС, можно гарантировать заданную длительность работы и период вызова периодических задач РВ.

Разработанное программное обеспечение, реализующее службу синхронизации процессов циклов РВ, представляет законченный продукт и используется без участия его разработчиков. Для удовлетворения этого требования выбрана модульная структура построения программного обеспечения. Модуль синхронизации процессов РВ непосредственно предназначен для реализации функциональности службы РВ. В состав этого модуля входят: менеджер задач, сами задачи РВ, модуль интерфейса с пользователем.

Менеджер задач является главной частью как модуля синхронизации процессов, так и всей службы РВ в целом. Приоритетный способ управления задачами РВ требует решения проблем, необходимых для создания менеджера:

- выбор базового класса приоритета службы РВ;
- определение количества задач, одновременно присутствующих в системе;
- выбор средства организации службы времени.

Выбор базового класса приоритета службы РВ очень важен, поскольку от него зависит правильность функционирования и количество одновременно присутствующих в системе задач. Как известно, Windows2000 предоставляет два базовых класса приоритета для задач РВ: классы high и real-time. Предполагается, что класс real-time является предпочтительным для задач РВ, поскольку: имеет наибольший уровень приоритетов потоков в системе, имеет четырнадцать относительных приоритетов потоков, обеспечивает наиболее точный вызов задач по времени. Но при использовании класса real-time в выбранной потоковой модели многозадачности существует и несколько препятствий:

- Пользователь утрачивает возможность руководить работой службы РВ, если менеджер задач будет иметь базовый класс приоритета real-time. При этом служба будет иметь приоритет выше, чем приоритет системной программы Explorer, обеспечивающей реакцию системы на действии пользователя, и вследствие этого система не сможет обрабатывать сообщения от драйверов.
- Система может не успевать выполнять свои внутренние функции, поскольку служба РВ постоянно будет занимать процессорное время и может произойти крах системы.

Таким образом, базовый класс приоритета real-time нецелесообразно использовать для менеджера задач РВ. Этот базовый класс можно использовать, если служба РВ будет построена по схеме многозадачности процессов. При этом менеджер задач и сами задачи должны реализовываться, как отдельные процессы системы, что даст возможность назначить задачам базовый класс real-time, а службе РВ обычный приоритет. Система будет работать корректно и пользователь сможет управлять задачами РВ, если задачи не будут занимать все процессорное время. Схема многозадачности процессов требует намного больше системных ресурсов, чем потоковая схема, а главное замедляет переключение контекста задач, поскольку каждая задача является отдельным процессом. Следовательно, использование этой схемы не эффективно. Базовый класс приоритета high также предназначен для задач РВ, но он имеет возможности меньшие, чем класс real-time, а именно: низший уровень приоритетов, работает в одном диапазоне уровней приоритетов с ОС,

имеет всего семь относительных приоритетов. При выполнении следующих ограничений его можно использовать для построения службы РВ: во время работы службы РВ не выполнять действий, которые могут привести к продолжительному функционированию системных сервисов, назначить менеджеру наибольший относительный приоритет. Таким образом, выбор базового класса приоритета high для модуля задач РВ полностью обеспечит правильную работу службы синхронизации процессов РВ. При использовании этого класса в системе могут одновременно функционировать шесть задач РВ.

Поскольку программная реализация имеет модульное построение, то задачи РВ размещаются в отдельном модуле, для предоставления пользователю возможности создавать собственные задачи. Менеджер задач разработан в виде модуля, разделенного на две части – центральный модуль управления задачами на приоритетной основе и модуль, встраиваемый в задачи РВ. Центральный модуль управления задачами на приоритетной основе является основной частью менеджера задач и выполняет следующие функции:

- загружает задачи, как отдельные потоки системы, в оперативную память в приостановленном виде;
- устанавливает для менеджера задач наибольший относительный приоритет;
- устанавливает приоритеты задачам в соответствии с периодами их вызова;
- запускает выполнение потоков задач РВ.

Менеджер устанавливает для себя наибольший относительный приоритет, что позволяет ему постоянно контролировать функционирование задач. Установка приоритетов задачам происходит в соответствии с периодами их вызова. То есть, задаче с наименьшим периодом устанавливается наибольший относительный приоритет после менеджера, а задаче с наибольшим периодом – наименьший приоритет. Это определяется тем, что задача с наименьшим периодом должна выполняться чаще, а поэтому она должна иметь наибольший приоритет, дающий ей возможность перерывать задачи с большим периодом. После отработки центрального модуля менеджера, задачи начинают функционировать в системе, а дальнейшее управление ими осуществляет часть менеджера, встраиваемая в каждую задачу. Модуль, включенный унифицировано в каждую задачу, является второй частью менеджера задач и предназначен для контроля длительности функционирования задачи. Каждая задача цикла РВ оформляется как бесконечный цикл, в конце каждой итерации которого вызовется модуль менеджера задач. Модуль определяет текущую продолжительность функционирования задачи с помощью службы времени, рассчитывает следующий момент времени вызова задачи и приостанавливает задачу до наступления этого момента. Так как в системе во время вызова задачи не будет присутствовать ни один поток с высшим приоритетом, то обеспечивается заданного время вызова потоков и стабильная длительность кванта работы каждого потока. Служба времени менеджера задач цикла РВ, с помощью функции *GetThreadTimes* устанавливает из объекта ядра потока текущее время функционирования потока в этом периоде вызова с дискретностью в 100 наносекунд, преобразовывает это время к форме с дискретностью 1 миллисекунда и возвращает его менеджеру задач для расчета следующего момента вызова задачи. Дискретность в 1 миллисекунду связана с тем, что для приостановки задач использована системная функция Sleep, которая имеет именно такую дискретность [3]. Служба синхронизации задач времени рассчитана на функционирование с модулем задач РВ, применяемым проектировщиком и имеет специфический характер для каждого объекта управления. Для этого служба РВ предоставляет все необходимые средства. Однако служба РВ выдвигает к задачам несколько требований. При формировании параметров задач необходимо придерживаться критерия [4]:

$$\sum_{i=1}^{N} \frac{\tau_i}{T_i} \le 1 \ , \qquad (3.1)$$

где N – количество задач;

$\tau_и$ – обеспечиваемая компьютером длительность работы текущей задачи;

Ti – требуемый период вызова текущей задачи.

Выполнение критерия обеспечивает параллельное функционирование задач в системе. Служба РВ разделяет общую длительность времени работы задач с большим периодом вызова, на несколько участков, для того чтобы уравновесить загрузку системы и обеспечить полное использование системных ресурсов. Число задач с разным периодом вызова не может быть больше шести. Такое ограничение выдвигается из-за лимитированного количества относительных приоритетов задач РВ в системе. Если пользователю требуется управлять большим количеством задач, то нужно свести обработку задач с близким периодом вызова в одну функцию. Это потребует решения расширенной задачи нелинейного целочисленного программирования [4], для построения эффективного расписания. Периоды вызова задач должны быть упорядочены по возрастанию. При этом задача 1 должна иметь наименьший период вызова, а задача 6 – наибольший. Для обмена данными между задачами необходимо использовать общие глобальные переменные модуля задач РВ, их количество и функции полностью зависят от пользователя. При выполнении всех изложенных требований, можно гарантировать правильное функционирование службы синхронизации.

Заключение

Испытания службы синхронизации процессов проведены на тестовых задачах. Каждая задача выполняла запись в общий файл результатов признака своего вызова. Практические испытания показали полную работоспособность службы синхронизации процессов РВ и доказали обоснованность выбора приоритетного метода управление задачами.

Литература

1. Жданов А. А. Операционные системы реального времени. – М: «PC Week» №20, 1999.
2. Калядин А. Windows NT для встраиваемых приложений“ Открытые системы”, №2, 1998г.
3. Рихтер Дж. Windows для профессионалов / Пер. с англ. – 4-е изд. – СПб.: Питер, 2001. – 752 с., ил.
4. Иванов А. Ю. Оптимизация планирования цикла реального времени вычислительной системы. Сборник трудов ДонГТУ. – Донецк; ДонГТУ, 1997. – 10 с., ил.
5. Чернобровцев А. Фрагменты реального времени. Еженедельник “Computerworld”, № 10, 2002г.
6. Кирюхин П. RTX – расширение реального времени для Windows NT. – http://www.citforum.ru

Поступила в редакційну колегію 29.03.2002