**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра «Информационные системы»**

Реферат №4

**по дисциплине «Системы реального времени»**

Тема: **Особенности операционных систем реального времени (ОСРВ)**

|  |  |  |
| --- | --- | --- |
| Студентгр. 9373 |  | Блохина А.А. |
| Преподаватель |  | Сидельников В.В. |

Санкт-Петербург

2023

**содержание**

[1. Требования к ОСРВ 3](#_Toc130566598)

[2. Особенности архитектуры 6](#_Toc130566599)

[3. Стандарт POSIX, расширения реального времени. 9](#_Toc130566600)

[4. Проблема инверсии приоритетов. 11](#_Toc130566601)

[5. Протокол наследования приоритетов, протокол граничных приоритетов. 11](#_Toc130566602)

[6. Инверсия приоритетов и мьютекс 13](#_Toc130566603)

# Требования к ОСРВ

Базовые требования, предъявляемые к системам реального времени:

* Своевременная реакция. После того как произошло событие, реакция должна последовать не позднее, чем через требуемое время. Превышение этого времени рассматривается как серьезная ошибка.
* Одновременная обработка информации, которая характеризует изменение процесса нескольких событий. Даже если одновременно происходит несколько событий, реакция ни на одно из них не должна запаздывать. Это означает, что система реального времени должна иметь встроенный параллелизм. Параллелизм достигается использованием нескольких процессоров в системе и/или многозадачным подходом.

Определим операционную систему реального времени как операционную систему, с помощью которой можно построить систему жесткого реального времени. Обязательные требования к ОСРВ:

* ОСРВ должна быть многонитиевой или многозадачной и поддерживать диспетчеризацию с вытеснением. ОСРВ должна обладать возможностью многонитевой или многозадачной работы и поддерживать диспетчеризацию с вытеснением, чтобы гарантировать предсказуемое поведение. Это означает, что максимальное время, необходимое для выполнения любой операции, должно быть известно заранее и соответствовать требованиям приложения. Например, система Windows 3.11 не подходит для построения систем реального времени, так как одно приложение может захватить управление и заблокировать все остальные. Для обеспечения вытеснения нитей и передачи управления наиболее приоритетной нити, планировщик должен иметь возможность вытеснения любой нити. Для обеспечения этой функциональности на уровне прерываний необходима многоуровневая структура обслуживания прерываний, включая аппаратную архитектуру.
* Приоритеты задач. В системе реального времени должны существовать гарантии того, что событие с высоким приоритетом будет обработано перед событием более низкого приоритета.

ОСРВ должна обладать развитой системой приоритетов. Во-первых, это требуется потому, что система сама может рассматриваться как набор приложений, подразделяющихся на потоки, и несколько высоких уровней приоритетов должны быть выделены системным процессам и потокам. Во-вторых, в сложных приложениях необходимо все потоки реального времени помещать на разные приоритетные уровни, а потоки не реального времени помещать на один уровень (ниже, чем любые потоки реального времени). При этом потоки не реального времени можно обрабатывать в режиме циклического планирования, при котором каждому процессу предоставляется квант времени процессора, а когда квант заканчивается, контекст процесса сохраняется, и он ставится в конец очереди. Во многих ОСРВ для планирования задач на одном уровне используется режим циклического планирования.

* Наследование приоритетов. Должна существовать система наследования приоритетов.

Комбинация приоритетов потоков и разделение ресурсов между ними приводит к проблеме инверсии приоритетов. Это можно проиллюстрировать на примере, где есть как минимум три потока. Когда поток низшего приоритета захватил ресурс, разделяемый с потоком высшего приоритета, и начал выполняться поток среднего приоритета, выполнение потока высшего приоритета будет приостановлено, пока не освободится ресурс и не отработает поток среднего приоритета. В этой ситуации время, необходимое для завершения потока высшего приоритета, зависит от нижних уровней приоритетов — это и есть инверсия приоритетов. В такой ситуации трудно выдержать ограничение на время исполнения.

Чтобы устранить такие инверсии, ОСРВ должна допускать наследование приоритета, то есть повышение уровня приоритета потока до уровня потока, который его вызывает. Наследование означает, что блокирующий ресурс поток наследует приоритет потока, который он блокирует, но это справедливо только в случае, когда блокируемый поток имеет более высокий приоритет.

* Предсказуемость. Поведение операционной системы должно быть известно и достаточно точно прогнозируемо. Время выполнения системных вызовов и временные характеристики поведения системы в различных обстоятельствах должны быть известны разработчику. Поэтому создатель ОСРВ должен приводить следующие характеристики:
* задержку прерывания, то есть время от момента прерывания до момента запуска задачи: она должна быть предсказуема и согласована с требованиями приложения;
* максимальное время выполнения каждого системного вызова (оно должно быть предсказуемо и не должно зависеть от числа объектов в системе).
* Управление памятью в системах реального времени (СРВ) играет важную роль в проектировании таких систем. На производительность и эффективность работы СРВ влияет как сложность программы управления, так и характеристики компьютера, связанные с подсистемой прерываний. Внешние события в СРВ становятся известны системе посредством прерываний, и важными характеристиками компьютера в таких случаях являются наличие как можно большего количества уровней прерываний и как можно меньшее время реакции на прерывание. СРВ также может сама являться инициатором периодических процессов, поэтому необходимо иметь в наличии один или несколько таймеров, которые могут работать в периодическом или ждущем режиме. С учетом того, что СРВ часто управляет ответственными промышленными процессами, требования к надежности используемого оборудования очень жесткие. Основными потребителями СРВ были военная и космическая области, но сейчас СРВ можно встретить даже в товарах широкого потребления. Применение СРВ распространяется на бортовое и встраиваемое оборудование, системы измерения и управления, радары, цифровые видеосистемы, симуляторы, системы управления производством и технологическим процессом, автомобили, энергетические системы, телекоммуникации и банковское оборудование.

# Особенности архитектуры

Системы реального времени отличаются от систем общего назначения тем, что у них четко разделены система разработки и система исполнения. Система исполнения - это набор инструментов, включая ядро, драйверы и исполняемые модули, обеспечивающих работу приложения реального времени на целевой системе, которая может быть различных аппаратных архитектур, таких как Intel, Motorola, RISC, MIPS, PowerPC и другие. Системы разработки обычно работают в популярных ОС, таких как UNIX, но могут также включать резидентные средства разработки, которые исполняются в операционной системе реального времени. Средства разработки для систем реального времени часто содержат инструменты удаленной отладки, профилирования, эмуляции целевого процессора, а также средства моделирования.

***Монолитная архитектура****.*

Монолитная архитектура - структура системы, где операционная система написана в виде процедур с определенными интерфейсами, которые могут вызывать друг друга. Для построения такой системы все процедуры компилируются и связываются в единый объектный файл. Хотя каждая процедура может видеть другую, существует некоторая структура в виде главной программы, служебных процедур, выполняющих системные вызовы, и утилит, обслуживающих служебные процедуры. При обращении к системным вызовам параметры помещаются в определенные места, после чего выполняется команда вызова ядра, переключающая машину в режим ядра и передающая управление операционной системе. Операционная система проверяет параметры вызова, определяет, какой системный вызов должен быть выполнен, и обращается к таблице с номером системного вызова в качестве индекса, в k-м элементе которой содержится ссылка на процедуру обработки системного вызова.

Объектно-ориентированный подход применяется для создания приложений, наилучшим образом поддерживающих этот подход. Операционные системы реального времени же используют процедурный подход. Сочетание объектно-ориентированных приложений и процедурных операционных систем имеет ряд недостатков, включая разрыв парадигмы программирования, неиспользование возможностей объектно-ориентированного подхода и потери производительности.

Строить операционную систему, используя объектно-ориентированный подход, имеет следующие преимущества:

* как приложение, так и операционная система полностью объектно-ориентированны и используют все преимущества этого подхода;
* приложение и операционная система могут быть полностью интегрированы, поскольку используют один объектно-ориентированный язык программирования;
* обеспечивается согласование интерфейсов операционной системы и приложения;
* приложение может моделировать операционную систему для своих потребностей, заказывая нужные объекты;
* единый комплекс (приложение + операционная система) является модульным и легко модернизируемым.

Основное преимущество монолитной архитектуры – относительная быстрота работы. Недостатки монолитной архитектуры включают увеличение времени работы системных вызовов, требующих переключения уровней привилегий, и риск того, что высокоприоритетная задача может не получить управления из-за работы низкоприоритетной задачи.

***Модульная архитектура на основе микроядра.***

Модульная архитектура появилась, как попытка убрать узкое место API и облегчить модернизацию системы и перенос ее на новые процессоры.

API в модульной архитектуре играет только одну роль: обеспечивает связь прикладных процессов и специального модуля менеджера процессов. Однако теперь микроядро играет двойную роль:

* управление взаимодействием частей системы (например, менеджеров процессов и файлов);
* обеспечение непрерывности выполнения кода системы (отсутствие переключения задач во время исполнения микроядра).

Недостатки модульной архитектуры фактически те же, что и у монолитной архитектуры. Проблемы перешли с уровня API на уровень микроядра. Системный интерфейс по-прежнему не допускает переключения задач во время работы микроядра, только сократилось время пребывания в этом состоянии. API по-прежнему может быть реализован только на ассемблере, проблемы с переносимостью микроядра уменьшились (в связи с сокращением его размера).

***Объектная архитектура на основе объектов-микроядер.***

Объектная архитектура на основе объектов-микроядер обеспечивает возможность переключения задач в любое время, модульность, безопасность, легкость модернизации и повторного использования кода благодаря объектно-ориентированному подходу. В этой архитектуре отсутствует API, и взаимодействие между компонентами системы и пользовательскими процессами осуществляется через обычный вызов функций на одном языке (C++), что обеспечивает максимальную скорость системных вызовов. Компилятор и динамический редактор объектных связей (linker) играют роль API. При старте приложения динамический linker загружает нужные ему микроядра, что позволяет сократить объем требуемой памяти. Система не использует виртуальную память, что ускоряет ее работу. Приложения и микроядра не зависят от начального адреса, что обеспечивает возможность их записи и исполнения в ПЗУ и оперативной памяти. Микроядра по своим характеристикам напоминают структуры, используемые в других операционных системах, но имеют свои различия. ОСРВ SoftKernel использует эту архитектуру.

# Стандарт POSIX, расширения реального времени.

Стандарт POSIX 1003.1 определяет СРВ следующим образом: «Реальное время в операционных системах - это способность операционной системы обеспечить требуемый уровень сервиса в заданный промежуток времени».

***Расширения реального времени для Windows NT***

Расширения реального времени для Windows NT были созданы несколькими фирмами после появления операционной системы Windows NT. Это свидетельствует о том, что такие продукты были востребованы на рынке. Расширения реального времени позволяют получить в системе Windows NT все возможности, такие как огромный набор прикладных программ, мощный программный интерфейс WIN32 и большое количество специалистов, знающих эту систему.

Хотя Windows NT создавалась как сетевая операционная система, она имеет элементы реального времени, такие как двухуровневую систему обработки прерываний (ISR и DPC) и классы реального времени. Однако, Windows NT не годится для построения систем жесткого реального времени из-за непредсказуемости системы, величины системы и отсутствия механизмов защиты от зависаний. Поэтому, даже в системах мягкого реального времени, Windows NT может быть использована только при выполнении целого ряда рекомендаций и ограничений.

Разработчики расширений пошли двумя путями. Они использовали ядра классических операционных систем реального времени в качестве дополнения к ядру Windows NT, например, фирмы "LP Elektroniks" и "Radisys". В этом случае параллельно с Windows NT работает операционная система VxWorks или InTime. Кроме того, предоставляется набор функций для связи приложений реального времени и приложений Windows NT. Технология использования двух систем на одном компьютере следующая: работу с объектом выполняет приложение реального времени, передавая затем результаты приложениям Windows NT для обработки, передачи в сеть и т.д.

Фирма VenturCom предлагает расширения реального времени для Windows NT, которые интегрируют реальное время в операционную систему. Они базируются на модификациях уровня аппаратных абстракций Windows NT, и включают подсистему реального времени RTSS и дополнительный набор объектов, управляемых специальным планировщиком реального времени. Решения VenturCom обеспечивают стабильность и надежность системы, а также простоту создания программ управления устройствами. Они также предоставляют возможность конфигурирования Windows NT и создания встроенных конфигураций. Результаты тестирования показывают, что эти продукты могут быть использованы для построения систем жесткого реального времени в будущем. Расширения реального времени VenturCom предназначены для больших систем реального времени, включая визуализацию, работу с базами данных и доступ в Интернет.

# Проблема инверсии приоритетов.

Инверсия приоритетов - это проблема, которая может возникнуть при использовании нескольких нитей в программе. Она заключается в изменении обычного порядка приоритетов нитей, что может привести к задержкам выполнения высокоприоритетных задач.

Для создания условия инверсии приоритетов должно быть задействовано как минимум три нити, и если нить с самым низким приоритетом заблокировала ресурс, который она делит с самой высокоприоритетной нитью, то время, необходимое для завершения нити с наивысшим приоритетом, может зависеть от времени работы нити с более низким приоритетом.

Для предотвращения инверсии приоритетов необходимо использовать механизм наследования приоритетов и защитить систему от изменения их порядка. Также необходимо правильно комбинировать приоритеты нитей и разделять между ними ресурсы.

# Протокол наследования приоритетов, протокол граничных приоритетов.

Протокол наследования приоритетов - это механизм, который позволяет процессу наследовать приоритет родительского процесса или потока. Этот механизм широко используется в многозадачных операционных системах, чтобы обеспечить более эффективное использование ресурсов процессора и снизить вероятность блокировки процессов с более высоким приоритетом.

В протоколе наследования приоритетов каждый процесс имеет свой приоритет, который определяет, как часто процесс будет выполняться относительно других процессов в системе. Когда процесс порождает новый процесс или поток, новый процесс или поток наследует приоритет своего родительского процесса или потока. Если родительский процесс имеет более высокий приоритет, чем дочерний процесс или поток, то он может установить приоритет дочернего процесса или потока равным своему собственному приоритету.

Протокол наследования приоритетов помогает оптимизировать использование ресурсов процессора в системе, позволяя процессам, которые выполняются в настоящее время, сохранять свой приоритет и предотвращая блокировку процессов с более высоким приоритетом. Однако неправильное использование этого механизма может привести к ситуации, когда процессы с более низким приоритетом блокируют процессы с более высоким приоритетом, что может привести к замедлению работы системы.

Протокол граничных приоритетов - это метод управления приоритетами в реальном времени, который определяет границы приоритетов для задач в системе.

Каждая задача имеет свой уровень приоритета, и протокол граничных приоритетов определяет нижнюю и верхнюю границы этого приоритета. Задачи с более высоким приоритетом не могут прервать задачи с более низким приоритетом, если их приоритет находится в пределах границ, установленных для низкоприоритетной задачи.

Протокол граничных приоритетов позволяет управлять приоритетами в системе реального времени и обеспечивать предсказуемость поведения системы. Это особенно важно для систем, в которых необходимо избежать непредсказуемых задержек и сбоев, например, в системах автоматического управления.

Протокол граничных приоритетов может использоваться в различных системах реального времени, таких как операционные системы, промышленные контроллеры, системы автоматического управления и т.д.

# Инверсия приоритетов и мьютекс

Для предотвращения инверсии приоритетов используются мьютексы. Мьютекс – это средство синхронизации, которое используется для предотвращения доступа к ресурсу из нескольких потоков одновременно. Когда поток хочет получить доступ к защищенному ресурсу, он должен сначала запросить мьютекс. Если мьютекс свободен, то поток получает его и может безопасно использовать ресурс. Если мьютекс уже занят другим потоком, то поток, запрашивающий мьютекс, блокируется до тех пор, пока мьютекс не станет доступен.

Таким образом, при использовании мьютексов задача с более высоким приоритетом будет блокировать задачу с более низким приоритетом только тогда, когда она захочет получить доступ к заблокированному ресурсу. В остальных случаях задача с более высоким приоритетом будет продолжать работу, не блокируя задачу с более низким приоритетом. Таким образом, инверсия приоритетов будет предотвращена, и задачи будут работать корректно и эффективно.

**Список использованной литературы:**

* + 1. Сергей Сорокин. Системы Реального Времени. // СТА. – 1997. - №2. – С 22-29.
    2. IEEE Standard Glossary of Software Engineering Terminology. / New York : The Institute of Electrical and Electronics Engineers, 1990. – URL: http://www.informatik.htw-dresden.de/~hauptman/SEI/IEEE\_Standard\_Glossary\_of\_Software\_Engineering\_Terminology%20.pdf
    3. Martin Timmerman, Bart Van Beneden, Lourent Uhres. RTOS Evaluation Kick Off! // Real-Time Magazine. – 1998. – N3 pp 6 – 10.
    4. Сидельников В.В. Лекции по дисциплине «системы реального времени». СПб.: Изд-во СПбГЭТУ «ЛЭТИ»
    5. Системное программное обеспечение управляющих систем реального времени: учебное пособие / М.В. Кавалеров. – Пермь: Изд-во ПНИПУ, 2013. – 190 с.
    6. Гриценко Ю.Б. Системы реального времени: Учебное пособие. — Томск: Томский межвузовский центр дистанционного образования, 2009. — 256 с
    7. Операционные системы реального времени: учебное пособие / Егор Горошко. – Харьков: Изд-во ИРЭ НАН Украины, 2003.
    8. Гордеев А.В. Операционные системы. 2-е изд. / СПб.: Питер, 2004. – 415с.: ил.
    9. Бурдонов И. Б., Косачев А. С., Пономаренко В. Н. Операционные системы реального времени/ Институт системного программирования РАН, 2006. – 98с.: ил.