Различение сегментов разных приложений

Для различения сегментов разных приложений используется таблица сегментов (ТС), которая содержит записи для каждого сегмента приложения. Адрес ТС хранится в аппаратном регистре таблицы сегментов (РТС) каждого процессора. В каждой строке ТС содержится следующая информация:

1. **Признак присутствия сегмента в ОЗУ (Пр)**
2. **Адрес (Базовый регистр) сегмента (Адрес)**
3. **Длина (Граница) сегмента (L)**
4. **Биты защиты для контроля доступа к сегменту (БЗ)**

Для доступа к слову [S,d] выполняются следующие шаги:

1. По РТС осуществляется обращение к ТС.
2. Проверяется присутствие сегмента в памяти (по признаку Пр).
3. При наличии сегмента в ОЗУ в строке ТС выбирается адрес сегмента.
4. В ячейке выбирается искомое слово.

Если сегмента в ОЗУ нет, происходит прерывание из-за отсутствия сегмента в памяти.

Переключение системы на другую программу (приложение) сводится к замене содержимого РТС на адрес другой таблицы. Сегменты могут размещаться в несмежных областях памяти, и не все сегменты должны одновременно находиться в памяти (часть может располагаться на внешних устройствах). Сегментация облегчает организацию разделения параллельно используемых программ, так как информация о параллельно используемой программе вносится в несколько таблиц сегментов. Программа при этом обрабатывает разные сегменты данных, описанные в таблицах различных приложений.

Совместное использование кода программы

Совместное использование кода программы несколькими приложениями может быть реализовано следующими способами:

1. **На аппаратном уровне**
- В этом случае логика организации совместно используемой памяти и единого адресного пространства реализуется непосредственно в аппаратуре вычислительной системы.

2. **На уровне операционной системы**
- Операционная система обеспечивает управление совместно используемой памятью и адресным пространством, что позволяет нескольким приложениям использовать один и тот же код.

3. **На уровне отдельного приложения**
- В крупных приложениях может быть реализована собственная логика управления совместно используемой памятью и адресным пространством, что позволяет нескольким процессам или потокам использовать общий код.

Основной проблемой при этом является обеспечение непротиворечивости копий совместно

используемых и изменяемых объектов в различных узлах вычислителя. Для решения этой проблемы используются протоколы когерентности с различными моделями состоятельности.

Процесс записи данных

Запись данных происходит следующим образом:

1. **Удаление узла из списка:**
- Узел, который был элементом списка, удаляет себя из списка и запрашивает доступ к памяти с правом записи данных.

2. **Получение права на запись:**
- Узел становится в начале списка и получает право модифицировать строку.
- При этом он обязан очистить все последующие элементы списка.

3. **Поддержка последовательного выполнения:**
- При слабой поддержке запись может быть произведена до завершения очистки списка.
- При сильной поддержке запись производится только после очистки списка.

4. **Обмен данными между узлами:**
- Возможен обмен данными между двумя узлами, находящимися в кэше, без обращения к памяти до тех пор, пока третий узел не запросит эти данные.

Доступ к Адресуемому Объекту при Страничной Организации

1. **Обращение к таблице страниц:**
   - По логическому адресу [P, d] выполняется обращение к таблице страниц (ТСтр) с использованием регистра РТСтр.
   - В выбранной строке таблицы страниц (ТСтр) определяется адрес страницы.

   2. **Поиск слова в странице:**
   - В ячейке страницы, соответствующей смещению d, находится искомое слово.

   Если требуемая страница отсутствует в оперативной памяти (ОЗУ), происходит прерывание из-за отсутствия страницы, что требует её подкачки из внешней памяти.

   При использовании сегментно-страничной организации памяти, адрес представляется в виде тройки [S, P, d], где S - номер сегмента, P - номер страницы в сегменте, d - смещение внутри страницы. Для выбора адресуемого объекта при такой трёхкомпонентной ссылке необходимо выполнить три цикла обращения к памяти:

   1. **Обращение к таблице сегментов:**
   - Получение адреса таблицы страниц для сегмента S.

   2. **Обращение к таблице страниц:**
   - Получение адреса страницы P в сегменте S.

   3. **Обращение к странице:**
   - Получение значения адресуемого объекта по смещению d внутри страницы P.

Местоположение регистра

Адрес таблицы сегментов (ТС) хранится в аппаратном **регистре таблицы сегментов (РТС)** каждого процессора.

Условия тупика

Необходимые (но не достаточные) условия тупика:

1. Условие взаимоисключения: Процессы требуют монопольного управления выделенными им ресурсами.

2. Условие удержания и ожидания ресурсов: Процессы удерживают за собой уже выделенные ресурсы и ожидают выделения дополнительных ресурсов.

3. Условие неперераспределяемости ресурсов: Ресурсы нельзя отобрать у процессов, удерживающих их, пока они не будут использованы для завершения работы.

4. Условие кругового ожидания: Существует кольцевая цепь процессов, в которой каждый процесс удерживает за собой один или более ресурсов, требующихся следующему процессу цепи.

Круговое ожидание

Круговое ожидание приводит к возникновению тупика. Для возникновения тупика необходимо выполнение следующих условий:

1. **Условие взаимоисключения:** процессы требуют монопольного управления выделенными им ресурсами.
2. **Условие удержания и ожидания ресурсов:** процессы удерживают за собой уже выделенные ресурсы и ожидают выделения дополнительных ресурсов.
3. **Условие неперераспределяемости ресурсов:** ресурсы нельзя отобрать у процессов, удерживающих их, пока они не будут использованы для завершения работы.
4. **Условие кругового ожидания:** существует кольцевая цепь процессов, в которой каждый процесс удерживает за собой один или более ресурсов, требующихся следующему процессу цепи.

Область значений общего семафора

Область значений общего семафора S:

- **Pr**: количество конкурирующих процессов
- **Res**: количество единиц ресурса
- **Диапазон значений: -Pr <= S <= Res**

Примечание:

- При S = -количество конкурирующих процессов: возникает тупик.

Одно значение

Определение монитора

Монитор выполняет несколько ключевых функций в системе защиты и управления процессами:

Монитор защиты:
- Служит воротами для большого класса сходных объектов, таких как файлы, требующие соблюдения правил доступа.
- Контролирует законность прав доступа, проверяя ключи, предъявляемые субъектами, и устанавливая соответствие ключа и замка, связанного с объектом.

Монитор управления взаимодействием параллельных процессов:
- Включает разделяемую управляющую структуру данных для централизации управляющей информации.
- Обеспечивает набор функций доступа к структуре данных для управления синхронизацией и коммуникациями между процессами.

Программный монитор (СУФ):
- Выполняет роль ворот и может быть расширен для обеспечения защиты.
- Может иметь программно-техническую реализацию.

Мониторы являются важным элементом для обеспечения безопасности и эффективного управления процессами в операционных системах.

Применение мониторов синхронизации

Мониторы синхронизации применяются в следующих случаях:

1. Получение критических ресурсов:
- Процесс использует операцию ТР для получения критических ресурсов, локализованных внутри тела монитора. Процесс может указать через параметры несколько семафоров с различными видами ресурсов, набор событий или тайм-аут ожидания одного из ресурсов или событий.

2. Синхронизация взаимодействующих процессов:
- Синхронизация осуществляется через анализ и изменение полей таблицы синхронизации (ТС). В ТС представлены все синхронизируемые процессы, упорядоченные по убыванию приоритета.

3. Управление параллельными процессами:
- Монитор управления взаимодействием параллельных процессов включает разделяемую управляющую структуру данных и набор функций доступа к структуре данных для управления синхронизацией и коммуникациями между процессами.

4. Обеспечение временной синхронизации:
- Для временной синхронизации используется процесс CLOCK, который имеет бесконечный цикл работы и приоритет, равный приоритету процедур монитора. Процесс CLOCK уменьшает на единичный интервал ненулевые элементы столбца таймера ТС и переводит процессы с истекшими тайм-аутами в состояние готовности.

Ожидания при применении мониторов защиты

При применении мониторов защиты ожидается следующее:

1. Разделение управления защитой по группам объектов:
- Каждый монитор защиты служит воротами для большого класса сходных объектов, например, СУФ требует соблюдения правил доступа для класса объектов – файлы.

2. Контроль законности прав доступа:
- Объекты, субъекты и домены именуются уникальными идентификаторами, что позволяет монитору защиты контролировать законность прав доступа.

3. Управление статусом защиты:
- Статус защиты системы управляется правилами изменения элементов матрицы, включая атрибуты владельца, управления и признака копирования.

4. Гибкость и эффективность:
- Схема замок-ключ позволяет эффективно и гибко управлять доступом, изменяя списки битовых наборов (замков), связанных с объектами.

5. Аутентификация, авторизация и аудит:
- Защита объектов операционных систем включает аутентификацию, авторизацию и аудит, что позволяет идентифицировать субъекта, определять разрешенные действия и фиксировать события, связанные с доступом к защищаемым ресурсам.

Пример неделимой операции

Пример неделимой операции - это операция **Test and Set Lock (TSL)**. Эта операция имеет два параметра:

- **ОБЩ**: глобальная переменная, общая для процессов.
- **ЛОК**: локальная переменная процесса.

Операция выполняется в виде двух присваиваний:

1. **ПиУ(ЛОК, ОБЩ)**: ЛОК := ОБЩ
2. **ОБЩ := 1**

Значения переменной ОБЩ:

- **1**: блокирующее значение.
- **0**: разблокирующее значение.

Пример выполнения:

- **ПРОЦЕСС 1**:
- ЛОК 1 := 1
- do while (ЛОК 1 = 1)
- ПиУ (ЛОК 1, ОБЩ)
- <критический участок>
- ОБЩ := 0
- <оставшаяся часть процесса 1>

- **ПРОЦЕСС 2**:
- ЛОК 2 := 1
- do while (ЛОК 2 = 1)
- ПиУ (ЛОК 2, ОБЩ)
- <критический участок>
- ОБЩ := 0
- <оставшаяся часть процесса 2>

Основной недостаток таких приемов - активное ожидание входа в критический участок.

Бригадное планирование

Бригадное планирование представляет собой метод планирования, при котором группы связанных потоков (бригады) планируются как единое целое. Все члены бригады запускаются одновременно на разных процессорах в режиме разделения времени. Основные характеристики бригадного планирования включают:

- **Запуск потоков:** Все потоки бригады начинают и завершают свои временные интервалы (кванты времени) вместе.
- **Перепланирование:** В начале каждого кванта все процессоры перепланируются заново, и на каждом процессоре запускается новый поток одной из готовых к выполнению бригад.
- **Простои процессоров:** Если какой-либо поток блокируется, его процессор простаивает до конца кванта, в течение которого перепланирование не выполняется.

Основная идея бригадного планирования заключается в том, чтобы все потоки бригады работали по возможности вместе, что позволяет устранить многозадачность и минимизировать потери времени процессоров при блокировании потоков операциями синхронизации.

Система планирования процессора с обратной связью

Система планирования процессора с обратной связью обладает следующими характеристиками:

- **Обслуживание коротких процессов:** Обеспечивает лучшее обслуживание коротких

процессов по сравнению с круговоротом.
- **Отсутствие предварительной информации:** Не требует никакой предварительной информации о процессах.
- **Обработка трудоемких процессов:** Трудоемкие процессы быстро перемещаются в последнюю очередь и не мешают выполнению коротких процессов.

Практическое применение алгоритмов планирования:
1. **ОС Реального времени и Встроенные системы:** Используют планирование по наивысшему приоритету.
2. **ОС Разделения времени и Диалоговые системы:** Используют планирование по наивысшему приоритету и круговорот.
3. **ОС Пакетной обработки:** Используют планирование с обратной связью.

Недостатки методов низкоуровневой синхронизации

Методы низкоуровневой синхронизации имеют несколько недостатков:

1. Блокировка памяти:
- Алгоритм Деккера, используемый для синхронизации двух процессов, трудно обобщается на произвольное число процессов.

2. Неделимая операция ПРОВЕРИТЬ и УСТАНОВИТЬ (ПиУ):
- Операция Test and Set Lock (TSL) имеет два параметра: общая глобальная переменная и локальная переменная процесса. Значения переменной ОБЩ: 1 – блокирующее, 0 – разблокирующее. Это может усложнять управление синхронизацией.
АКТИВНОЕ ОЖИДАНИЕ НЕДОСТАТОК

3. Потенциальное узкое место ведущего процессора:
- При интенсивном потоке системных вызовов от большого числа ведомых процессоров ведущий процессор может стать узким местом. Такая модель работает при небольшом числе процессоров и обычно применяется во встроенных системах.

4. Необходимость синхронизации доступа конкурирующих процессоров:
- В симметричной мультипроцессорной обработке требуется синхронизация доступа конкурирующих процессоров к управляющим структурам, что усложняет код и требует высококвалифицированных специалистов для сопровождения.

5. Потеря времени процессоров при блокировании потоков:
- При блокировании потоков операциями синхронизации происходит существенная потеря времени процессоров. Решением данной проблемы является бригадное планирование, которое устраняет многозадачность.

Компонента для виртуальной памяти

Для реализации виртуальной памяти необходима **аппаратная поддержка**. Это включает в себя поддержку процессором аппаратных технологий виртуализации, таких как Intel VT (VT-x, Intel Virtualization Technology for x86) или AMD-V. Аппаратная поддержка виртуализации позволяет реализовать в многопроцессорных вычислительных системах

доменную архитектуру, которая обеспечивает независимую работу различных приложений.

Подсистемы для реализации виртуальной памяти

Для реализации виртуальной памяти необходимы следующие подсистемы:

1. **Аппаратный уровень**
- Включает поддержку процессором аппаратных технологий виртуализации, таких как Intel VT (VT-x) или AMD-V, что позволяет реализовать доменную архитектуру в многопроцессорных вычислительных системах.

2. **Уровень операционной системы**
- Операционная система должна поддерживать управление виртуальной памятью, включая обработку прерываний при отсутствии адресуемого объекта в локальной памяти и удовлетворение запросов через коммуникационную сеть.

3. **Уровень приложений**
- Крупные приложения могут иметь собственные механизмы управления виртуальной памятью, что позволяет оптимизировать использование ресурсов и повысить производительность.

Эти подсистемы обеспечивают эффективное управление виртуальной памятью, изоляцию вычислительных процессов и поддержку многозадачности.

Действия при промахе сегмента страницы в памяти

При промахе сегмента страницы в памяти необходимо выполнить следующие действия:

1. **Проверка наличия сегмента в памяти:**
- Сначала проверяется присутствие сегмента в оперативной памяти (ОЗУ) по признаку Пр.
- Если сегмент отсутствует в ОЗУ, происходит прерывание из-за отсутствия сегмента в памяти.

2. **Загрузка сегмента:**
- При отсутствии сегмента в ОЗУ, система инициирует загрузку сегмента из внешней памяти (ВнУ) в ОЗУ.
- После загрузки сегмента в ОЗУ, система обновляет таблицу сегментов и таблицу страниц, чтобы отразить новое местоположение сегмента.

3. **Обращение к сегменту:**
- После успешной загрузки сегмента в ОЗУ, система повторяет обращение к требуемому адресу внутри сегмента.
- В случае успешного нахождения сегмента в ОЗУ, выбирается адрес сегмента и искомое слово.

Эти шаги обеспечивают корректное обращение к памяти при промахе сегмента страницы, минимизируя задержки и обеспечивая целостность данных.

Когерентность

Когерентность в вычислительных системах относится к согласованности данных в памяти, особенно в многопроцессорных системах. Существует два основных типа когерентности: явная (программная) и неявная (аппаратная).

Явная (программная) когерентность:
- Программист использует специальные команды для работы с локальной памятью и управления контроллерами каналов коммуникационной среды.
- Основная задача разработчика - эффективное программирование параллельных процессов, совмещающих вычисления и передачу данных между узлами, минимизируя объем передаваемой информации.
- Протоколы когерентности памяти являются надстройкой над аппаратными средствами передачи сообщений, что может создавать проблемы с пропускной способностью и миграцией данных.
- Программная реализация позволяет учитывать особенности прикладных программ и применять разнообразные протоколы согласованности данных.

Неявная (аппаратная) когерентность:
- Современные микропроцессоры имеют встроенные аппаратные средства обеспечения когерентности данных в иерархической памяти.
- Когерентность кэшей процессоров обеспечивается с помощью межузловых пересылок.
- Аппаратная реализация когерентности универсальна и не требует дополнительного кода в программах для организации разделяемой памяти.
- Обеспечивается минимальная задержка согласования копий данных и высокая пропускная способность всего механизма когерентности.

Организация виртуальной памяти

Для организации виртуальной памяти необходимо выполнить следующие шаги:

1. **Создание иллюзии полного размещения приложения в ОЗУ:**
- Каждый процесс должен иметь иллюзию, что все приложение находится в оперативной памяти (ОЗУ). При обращении процесса к информации, которой нет в ОЗУ, происходит прерывание, и супервизор перемещает нужную информацию с внешнего устройства в ОЗУ, предоставляя процессу доступ к ней.

2. **Многоуровневая организация памяти:**
- Виртуальная память организуется в виде многоуровневой памяти. Каждый уровень памяти (M(i)) имеет более высокую скорость доступа, меньшую емкость и более высокую удельную стоимость хранения по сравнению с уровнем M(j) при i<j. Объекты, находящиеся в памяти i-го уровня, также хранятся на уровнях с большими номерами. Если процессор обращается к объекту, которого нет в памяти 1-го уровня, система ищет его на уровнях с большими номерами и перемещает на верхние уровни вплоть до 1-го.

3. **Использование страниц фиксированного размера:**
- Многоуровневая память обычно разбита на страницы фиксированного размера. Это позволяет эффективно управлять памятью и минимизировать накладные расходы на управление памятью.

4. **Реализация логики организации памяти:**
- Логика организации совместно используемой памяти и единого адресного пространства может быть реализована на аппаратном уровне, на уровне операционной системы или на уровне отдельного крупного приложения.

Действия при отсутствии сегментов в памяти

Если сегмента нет в оперативной памяти (ОЗУ), происходит прерывание из-за отсутствия сегмента в памяти.

Основные шаги при доступе к слову [S,d]:
1. По регистру таблицы сегментов (РТС) выполняется обращение к таблице сегментов (ТС).
2. Проверяется присутствие сегмента в памяти по признаку Пр.
3. При наличии сегмента в ОЗУ в S-й строке выбирается адрес сегмента.
4. В d-й ячейке выбирается искомое слово.

Если сегмента в ОЗУ нет, происходит прерывание из-за отсутствия сегмента в памяти.

Случаи уплотнения памяти

Уплотнение памяти имеет смысл и происходит в следующих случаях:

Когда уплотнение выгодно:
- **Запросы на память велики по размеру** и из-за внешней фрагментации образуется большой процент пустот.
- **ЦП не занят выполнением активных заданий (простаивает)** и его можно занять уплотнением, что делает операцию бесплатной.

Действия при запросе области памяти, превосходящей наибольшую пустоту:
- **Отказ в удовлетворении запроса.**
- **Вытеснение какого-либо сегмента.**
- **Уплотнение** – перегруппировка

Состояния процесса

(готов к работе, работающий, заблокирован).

Процесс может находиться в следующих состояниях:

Глобальные состояния:
1. **NC (не кэшированная)** - если копия строки не находится в кэше какого-либо другого узла, кроме, возможно, резидентного для этой строки.
2. **RS (удаленно-разделенная)** - если копии строки размещены в кэшах других узлов.
3. **RM (удаленно-измененная)** - если строка изменена операцией записи в каком-либо

узле.

Локальные состояния:
1. **NU (невозможная к использованию)**.
2. **S (разделяемая)** - если есть неизмененная копия, которая, возможно, размещается в других кэшах.
3. **M (измененная)** - если копия изменена операцией записи.

Синхронизация с помощью монитора

Синхронизация процессов с помощью монитора осуществляется через использование процедур, которые анализируют и изменяют поля таблицы синхронизации (ТС). Основные аспекты синхронизации с помощью монитора включают:

Функции управления и синхронизации:
- **TP:** Процесс использует эту операцию для получения критических ресурсов, локализованных внутри тела монитора. Процесс может указать несколько семафоров с различными видами ресурсов, набор событий или тайм-аут ожидания одного из ресурсов или событий.
- **TV:** Операция, связанная с изменением состояния процесса.
- **POST и WAIT:** Процедуры, которые управляют состоянием процессов и их взаимодействием с ресурсами и событиями.

Логика работы монитора:
1. Синхронизация процессов осуществляется через анализ и изменение полей ТС.
2. В ТС представлены все синхронизируемые процессы, упорядоченные по убыванию приоритета.
3. Наименьший приоритет имеет вспомогательный процесс, который устраняет тупиковые ситуации.
4. В строке каждого процесса содержатся адрес области сохранения, переменная состояния процесса, тайм-аут ожидания, ссылка на список ресурсов и события, которые могут привести к переключению процессов.

Пример реализации:
- **Поставщик:** Готовит сообщение, использует семафоры для синхронизации и отправляет сообщение.
- **Потребитель:** Получает сообщение, использует семафоры для синхронизации и обрабатывает сообщение.

Монитор позволяет реализовать сложные схемы межпроцессного взаимодействия и решить задачи параллельного ожидания ресурсов и событий. Управление взаимодействием процессов с помощью ТС имеет ясную логическую структуру и является наглядным и простым.

Мера справедливости

Мера справедливости определяется как дисперсия времени доступа или пропускной способности (D(T)min / D(C)min). Этот показатель оценивает, насколько значения операционной характеристики отдельных запросов могут отклоняться от среднего

значения показателя. Если не использовать этот показатель, некоторые запросы могут ждать обслуживания неопределенно долго, а некоторые полностью игнорироваться.

Попали на таблицу сегментов – по регистру

Дальше в таблицу страниц

из чего состоит программа при сегментной организации?

Пространство адресов программы разделяется на отрезки – сегменты различной длины, соответствующие содержательно различным частям программы.

Каждые сегменты из страниц

В каких состояниях может находиться процесс? (готов к работе, работающий, заблокирован).

Что можно заказать у монитора ресурсов и где он обрабатывается?

У монитора ресурсов можно заказать следующие элементы:

1. **Критические ресурсы**:
- Процесс может указать несколько семафоров с различными видами ресурсов, одним из которых процесс хотел бы обладать.
- Набор событий, наступления одного из которых ожидает процесс.
- Тайм-аут ожидания одного из ресурсов или событий.

2. **Освободившиеся ресурсы**:
- Процесс возвращает монитору освободившийся ресурс.
- Поиск процесса, ожидающего этот ресурс, осуществляется сканированием столбца ТС РЕСУРСЫ и просмотром списков ожидаемых каждым процессом ресурсов.

Обработка заказов:
- Заказы обрабатываются внутри тела монитора, который анализирует и изменяет поля таблицы синхронизации (ТС).
- Номер строки текущего процесса присваивается переменной монитора ТЕКУЩИЙ. Текущим может стать только самый приоритетный процесс из всех готовых к работе.