

Средства взаимодействия процессов и потоков

Проблемы в многозадачных и параллельных системах:

Проблема взаимного исключения: Исключить выполнения действий, которые не должны происходить одновременно, например, при выводе документов на печать, где данные должны быть обработаны последовательно.

Проблема синхронизации: Обеспечение правильного порядка выполнения действий между потоками или процессами для корректного взаимодействия и передачи управления между ними.

Проблема обмена данными: вытекает из концепции изолированных адресных пространств – процессы не имеют возможности напрямую обращаться «внутри» друг друга. Потоки одного процесса действуют в едином адресном пространстве, но в общем случае во взаимодействии могут участвовать потоки разных процессов. Решением всех задач становится использование специальных системных объектов – объектов IPC (Inter-Process Communication) и, как частный случай, ISO (Inter-process Synchronization Objects)

Базовые понятия в параллельном программировании:

Критический ресурс – ресурс системы, который не может использоваться одновременно более чем заданным числом пользователей. Часто речь идет о доступности его не более чем одному пользователю.

Критическая секция – часть алгоритма, где происходят обращения к критическим ресурсам. Очевидно, число потоков, находящихся в критической секции, ограничивается в соответствии с характеристиками критического ресурса (часто – одним потоком).

Требования к критическим секциям: Ограничение количества потоков, ограничение времени пребывания, отсутствие приоритетов внутри секции, независимость от конкретных параметров системы.

IPC (Межпроцессное взаимодействие / Inter-Process Communication) - обмен данными между потоками одного или разных процессов.

Общая классификация IPC:

- **сигнальные** (signal): прерывания, сигналы, сообщения
- **канальные** (pipe): каналы (трубопроводы), почтовые ящики и сокеты различного вида
- **разделяемая память** (shared memory)
- **очереди сообщений** (message queue, MQ) – обладают свойствами сообщений и каналов
- **средства синхронизации** (ISO): семафоры, мьютексы, барьеры
- **комбинированные** (комплексные) решения

Критерии сравнения:

- **масштаб:** межпоточные, межпроцессные, межсистемные
- **функциональность:** синхронизация, обмен данными
- **быстродействие** (для синхронизации)

- **производительность**, пропускная способность (для данных)
- **универсальность**
- **трудоемкость**, уровень поддержки системой

Сигнальные IPC

Пригодны для извещения о событиях или условиях, а также для передачи простых команд. Передача существенных объемов данных затруднена или невозможна.

1. Прерывания (**interrupt**) и исключения (**exception**):

- Аппаратный уровень, обычно управляется операционной системой.
- Недоступен для прикладных программ.

2. Сигналы (**signal**):

- Предоставляют минимальную информацию: тип сигнала и идентификатор получателя.
- Могут быть аппаратными или программными.
- Часто используются в Unix-системах.
- В Windows их поддержка ограничена и используется редко (`signal()`, `raise()`).

3. Сообщения (**message**):

- Более информативны, могут содержать дополнительные параметры.
- Используются для передачи сложной информации и вызовов к другим процессам.
- В Windows существуют **оконные сообщения**, которые могут передаваться синхронно или асинхронно.
- Очереди сообщений поддерживают асинхронное взаимодействие и передачу потока данных между процессами.

ISO (средства синхронизации)

Механизмы синхронизации используются для управления доступом к общим ресурсам. Основная идея заключается в **проверке и изменении флага**, отражающего состояние ресурса (свободен или занят). Однако сами механизмы контроля доступа могут становиться **критическим ресурсом** и вызвать заикливание требований.

Для **обеспечения атомарности операций** с критическими секциями используются **привилегированные операции**, недоступные обычным процессам. Решением может быть использование **системных средств** для гарантии атомарности операций. Другой подход предполагает **временный запрет прерываний** во время доступа к критической секции, но такой метод требует привилегированных инструкций и может усложнить управление вложенными блокировками.

Простейшие решения для обеспечения атомарности включают использование **глобальных флагов** для гарантии правильного выполнения операций. Одним из таких решений является **алгоритм Деккера**.

Спин-блокировка (spinlock) - это метод, при котором процесс ожидает изменения значения переменной, не переходя в режим ожидания. Этот подход обеспечивает высокую скорость выполнения, так как не требуется переключение контекста для выхода из блокировки.

Однако основной недостаток спин-блокировки заключается в постоянной загрузке процессора, так как процесс активно проверяет переменную в цикле. Для выхода из спин-блокировки необходимо, чтобы значение переменной было изменено внешним образом, например, другим потоком или процессором, чтобы разблокировать ожидающий поток.

Семафоры, мьютексы и барьеры - это средства синхронизации в многозадачных системах.

Семафор – глобальная переменная-счётчик (S), которая позволяет управлять доступом к ресурсу с помощью двух операций:

P(S) (уменьшение): если счетчик больше 0, уменьшается на 1; если 0 – поток ждет.

V(S) (увеличение): счётчик всегда увеличивается на 1.

Семафоры позволяют синхронизировать потоки, блокируя их при недоступности ресурса.

Мьютекс – объект для обеспечения взаимного исключения, используется для контроля доступа к критическому ресурсу:

Имеет два состояния: **свободен** и **занят**.

Доступен только один поток одновременно; другие блокируются, пока мьютекс не освободится.

Мьютексы могут быть рекурсивными, что позволяет одному и тому же потоку повторно захватывать его.

Барьер – объект синхронизации, который блокирует потоки до достижения заданного количества запросов:

Используется для синхронного запуска потоков, когда все они достигли определенной точки.

Вариант реализации – счетчик запросов, например, основанный на семафоре.

Эти объекты применяются для координации потоков, особенно при работе с критическими секциями и синхронных задач.

"Блокированные" обращения - потокобезопасная (атомарная) системная реализация операций над переменными, также используется для межпроцессного взаимодействия. Поддерживаемые операции: InterlockedDecrement(), InterlockedIncrement(), InterlockedExchange().

Механизм Critical Section - упрощенный мьютекс для реализации критической секции в рамках одного процесса. Использует структуру CRITICAL_SECTION в адресном пространстве процесса. Функции API: InitializeCriticalSection(), DeleteCriticalSection(). Поддерживает рекурсивное вхождение в секцию.

Функции ожидания WaitFor... - применяются к системным объектам с флагом "signaled". Так, кроме специализированных объектов синхронизации это могут быть процессы, потоки, файлы и проч.

Типичная интерпретация состояния «signaled»:

- Файлы, каналы, устройства ввода-вывода: завершение операции ввода-вывода
 - Процессы, потоки: завершение (переход в состояние «зомби») либо прекращение действия дескриптора (Handle)
 - Объекты синхронизации – в соответствии с логикой конкретного вида объекта.
- WaitForSingleObject(), WaitForMultipleObjects(), SignalObjectAndWait(),

WaitFor***Ex(), с возможностью указания callback-функций для обработки завершения операций.

Объект Mutex - системный ISO-объект, реализующий классический мьютекс. Имеет уникальное имя и дескриптор (Handle), может использоваться в рамках одного процесса или между потоками разных процессов.

Состояния:

- «signaled» – мьютекс «свободен»
- «not signaled» – мьютекс «занят/захвачен»

Поддерживает повторный захват одним потоком. Создание с помощью CreateMutex(), открытие по имени с OpenMutex(), отказ от объекта CloseHandle(), освобождение с ReleaseMutex(), и захват с функцией ожидания.

Объект Semaphore - системный ISO-объект, соответствует классическому семафору. Идентифицируется уникальным именем и дескриптором (Handle), может использоваться в разных процессах. Состояния: "signaled" (значение больше 0) и "not signaled" (значение равно 0). Создание через CreateSemaphore(), открытие по имени с OpenSemaphore(), отказ от объекта CloseHandle(), Безусловный инкремент счетчика семафора V(S) ReleaseSemaphore(), и блокирующий декремент с функцией ожидания.

Объект Event - системный ISO-объект, представляющий программное событие. Имеет уникальное имя и дескриптор (Handle), может использоваться в рамках одного процесса или между потоками разных процессов. Состояния: "signaled" (событие произошло и не обработано) и "not signaled" (событие не произошло или уже обработано). Уведомляет потоки, что произошло событие.

Типы событий:

С **автоматическим** сбросом, с **ручным** сбросом (manual reset).

Функции: CreateEvent(), SetEvent(), Явный сброс в "not signaled": ResetEvent()

Временная установка в "signaled" до срабатывания всех ожидающих функций: PulseEvent()

Объект Waitable Timer - системный ISO-объект, таймер с несколькими параметрами и несколькими путями использования. Идентифицируется уникальным именем и дескриптором (Handle), может использоваться в разных процессах. Уведомляет потоки о получении указанного времени.

Типы таймеров:

– **manual-reset** – сброс таймера путем новой установки величины задержки явным вызовом функцией SetWaitableTimer()

– **synchronization** – сброс таймера автоматически при завершении Wait-функции

– **periodic manual reset**, **periodic synchronization** – аналогично, но перевод таймера в состояние «signaled» повторяется периодически через заданные интервалы времени.

Функции: CreateWaitableTimer(), OpenWaitableTimer(), CloseHandle(), SetWaitableTimer(), CancelWaitableTimer().

Канальные IPC: каналы и «почтовые ящики»

Каналы — это объекты, которые работают аналогично файлам и обеспечивают межпроцессное взаимодействие.

Типы каналов:

Неименованный канал (pipe):

- Используется только между родственными процессами.
- Не может быть открыт сторонними процессами.

Именованный канал (FIFO):

- Полноценный объект файловой системы.
- Доступен всем процессам в системе.

Основное **отличие каналов от обычных файлов** – доступ только последовательный, причем чтение – разрушающее (данные удаляются после прочтения).

Обработка доступа:

Чтение:

- Непустой канал - чтение данных.
- Пустой канал — блокировка (ожидание).
- Пустой канал без писателей — ошибка (для pipe).

Запись:

- Запись данных или ожидание места.
- Запись без читателей — ошибка (для pipe).

Ограничения:

- Нет встроенных средств для предотвращения коллизий.
- Параллельная запись может привести к смешиванию данных.
- Параллельное чтение может давать невалидные фрагменты.

Возможные решения:

- Ограничение участников взаимодействия.
- Отказ от двунаправленного обмена.
- Использование дополнительных средств синхронизации.

Почтовый ящик (mailslot)

Однонаправленность: Данные передаются только в одном направлении, что упрощает управление.

Идентификация: Каждый ящик связан с единственным «читателем», что снижает риск потери целостности данных.

Двунаправленный обмен:

Для двунаправленного обмена используются два однонаправленных почтовых ящика. (Минимизация коллизий: только один процесс читает данные. Упрощение идентификации: четкая связь между ящиком и читателем.)

Канальные IPC в Windows

Неименованные каналы (pipe)

Идентификация: Используется только дескриптор (Handle), нет имени.

CreatePipe(), CloseHandle()

Особенности:

- Два отдельных дескриптора: чтения и записи.
- Существуют только между родственными процессами.

- Удаляются после отключения последнего пользователя, данные теряются.

Именованные каналы (FIFO)

Идентификация: Дескрипторы для открытого канала и глобальные имена в файловой системе: `\\.\pipe\pipename`.

`CreateNamedPipe()`, `CloseHandle()`, `CreateFile()` - подключение к существующему каналу

Особенности:

- Несимметричность: выделяются клиент и сервер.
- Возможность нескольких экземпляров для разных клиентов.
- Непрочитанные данные теряются после закрытия экземпляра.
- Дескрипторы могут быть одно- или двунаправленными.
- Два типа: «byte-type» и «message-type» (управляется параметром `dwPipeMode`).
- Дополнительные функции: неразрушающее чтение, обратная связь.

Почтовые ящики (Mailslot)

Имена: `\\.\mailslot\slotname`, `\\computename\mailslot\slotname`, и т.д.

`CreateMailslot()`, `CloseHandle()`, `CreateFile()` - подключение к существующему ящику

Особенности:

- Владелец становится процесс, который создал ящик (доступ только для чтения).
- Может быть открыт удаленно через сеть.

Разделяемая память (Shared Memory) (В windows - отображение файла в память)

- Обеспечение: Совместный доступ к данным в выделенной памяти.
- Преимущества: Высокая производительность.
- Проблемы: Коллизии при параллельном доступе.
- Решение: Использование семафоров и мьютексов для синхронизации.

Очереди сообщений (Message Queue, MQ)

- Описание: Асинхронный механизм для передачи сообщений с буферизацией.
- Преимущества: Гарантия целостности сообщений.
- Примеры: Microsoft MQ, IBM MQ, Amazon SQS, Java Message Service.

Другие средства обмена данными в Windows

- Оконное сообщение: `WM_COPYDATA`
- Буфер обмена: `Clipboard`
- Глобальные данные: `DLL`
- Удаленный вызов процедур: `RPC`

Другие решения. Комплексные механизмы

Сокеты - механизм взаимодействия, близкий к каналам, используется в сетевом и локальном программировании.

Монитор Хоара - интерфейс для работы с общими данными, обеспечивает синхронизацию и скрывает доступ к критическим ресурсам.

Рандеву - концепция взаимодействия между процессами, требует предварительной подготовки для обмена данными. Этапы подготовки процессов:

- 1) заявить о намерении взаимодействовать (назначить рандеву)
- 2) подтвердить согласие на взаимодействие
- 3) объявить точку взаимодействия
- 4) подтвердить точку взаимодействия

Симметричное рандеву (Хоар) - процессы равноправны, операторы «!» и «?» обеспечивают передачу и прием данных.

Асимметричное рандеву (Хансен) - участники имеют заранее распределенные роли «хозяин» (посылает запросы) и «слуга» (принимает запросы), обмен данными осуществляется через вызов функций.

Эффективная реализация асимметричного рандеву требует поддержки на аппаратном уровне.

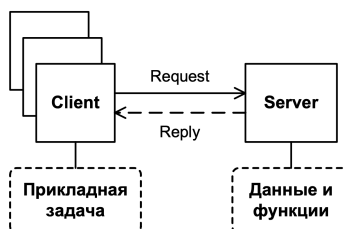
Развитие идеи асимметричного рандеву.

"Клиент-сервер":

Определение: Клиент обращается к серверу с запросом и получает ответ.

Характеристики: Предполагается, что серверов меньше, чем клиентов. Клиент может обращаться к нескольким серверам.

Пример: Клиент-серверные системы управления базами данных (СУБД).

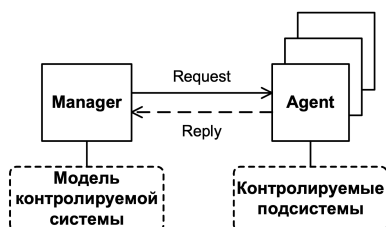


"Агент-менеджер":

Определение: Менеджер отправляет запросы множеству агентов, собирает ответы и управляет системой через агентов.

Характеристики: Предполагается, что агентов больше, чем менеджеров. Один агент может работать на нескольких менеджерах.


Примеры: Управление сетевыми протоколами и служба SNMP с базой MIB.



Объекты синхронизации

Объект синхронизации — это объект, дескриптор которого можно указать в одной из [функций ожидания](#) для координации выполнения нескольких потоков. Несколько процессов могут иметь дескриптор одного и того же объекта синхронизации, что делает возможной синхронизацию между процессами.

Следующие типы объектов предоставляются исключительно для синхронизации.

 Развернуть таблицу

Тип	Описание
Событие	Уведомляет один или более ожидающих потоков о том, что произошло событие. Дополнительные сведения см. в разделе Объекты событий .
Mutex	Может принадлежать только одному потоку за раз, что позволяет потокам координировать взаимноисключающий доступ к общему ресурсу. Дополнительные сведения см. в разделе Объекты мьютекса .
Semaphore	Поддерживает число от нуля до некоторого максимального значения, ограничивая количество потоков, которые одновременно обращаются к общему ресурсу. Дополнительные сведения см. в разделе Объекты семафора .
Таймер ожидания	Уведомляет один или несколько ожидающих потоков о получении указанного времени. Дополнительные сведения см. в разделе Объекты таймера для ожидания .