**Тема 4. Взаимодействие и планирование процессов**

**Лекция 13 «Элементарные средства межпроцессного взаимодействия. Подходы (методы) для межпроцессного взаимодействия».**

Межпроцессное взаимодействие (англ. inter-process communication, IPC) — обмен данными между потоками одного или разных процессов. Это набор программных интерфейсов, которые позволяют программисту координировать действия между различными программными процессами, которые могут одновременно выполняться в операционной системе. Это позволяет конкретной программе обрабатывать множество пользовательских запросов одновременно.

Процессам необходимо взаимодействовать между собой, поэтому необходимо правильно организовать взаимодействие между процессами (по возможности, не используя прерывания).

**К IPC относят:**

* передачу информации от одного процесса другому;
* контроль над деятельностью процессов: как гарантировать, что два процесса не пересекутся в критических ситуациях;
* согласование действий процессов: если процесс А должен поставлять данные, а процесс В выводить их на печать, то процесс В должен подождать и не начинать печатать, пока не поступят данные от процесса А.

**Механизмы, предоставляемые ОС и используемые для IPC:**

* механизмы обмена сообщениями;
* механизмы синхронизации;
* механизмы разделения памяти;
* механизмы удалённых вызовов (RPC).

Для оценки производительности различных механизмов IPC используют следующие параметры:

* пропускная способность (количество сообщений в единицу времени, которое ядро ОС или процесс способно обработать);
* задержки (время между отправкой сообщения одним потоком и его получением другим потоком).

Поскольку каждый запрос пользователя может привести к запуску нескольких процессов в операционной системе, процесс может потребовать взаимодействия друг с другом.

**Подходы (методы) для межпроцессного взаимодействия:**

1. **Трубопровод (канал, конвейер) -** это полудуплексный метод связи. Труба широко используется для связи между двумя взаимосвязанными процессами. Данные могут передаваться только в одном направлении, и его можно использовать только между связанными процессами. Родство процесса обычно относится к отношениям родительско-дочернего процесса.

Существует два типа каналов:

* **pipe (безымянный канал).** Пример канала: родительский процесс создает канал и записывает данные в канал, а дочерний процесс считывает данные из канала. Канал используется для связи между связанными процессами, такими как родительский и дочерний процессы. Он создается и открывается системным вызовом pipe (). Когда последний использующий его процесс закрывает ссылку на него, канал автоматически отменяется. Процесс может получить доступ только к каналам, созданным им самим или его предками, но не может получить доступ к существующим каналам, потому что у него нет имени.
* **FIFO (именованный канал).** Основное отличие от безымянных каналов состоит в том, что именованные каналы имеют имя, которое соответствует узлу индекса диска. С этим именем файла любой процесс с соответствующими разрешениями может получить к нему доступ. После создания любой процесс может открыть его, прочитать и записать его через имя файла, не ограничиваясь родительско-дочерним процессом (при условии, что процесс имеет надлежащий доступ к FIFO). Когда он больше не используется процессом, FIFO освобождается в памяти, но дисковый узел все еще существует.
* **2. Именованный канал FIFO -** является полудуплексным методом связи, но позволяет взаимодействовать между несвязанными процессами (первый процесс может взаимодействовать со вторым процессом, и наоборот).

Именованные каналы имеют следующие отличия:

* Именованные каналы существуют в виде специального файла устройства в файловой системе.
* Процессы различного происхождения могут разделять данные через такой канал.
* Именованный канал остается в файловой системе для дальнейшего использования и после того, как весь ввод/вывод сделан.

Чтобы физически открыть проход к каналу, должен быть использован системный вызов «open» или библиотечная функция. С полудуплексными каналами это невозможно, поскольку канал находится в ядре, а не в физической файловой системе. Если FIFO открыт для чтения, процесс его блокирует до тех пор, пока какой-нибудь другой процесс не откроет FIFO для записи.

1. **Очереди сообщений** – это связанный список сообщений, хранящихся в ядре. Идентифицируется по идентификатору очереди сообщений. Этот метод обеспечивает связь между одним или несколькими процессами с полнодуплексной пропускной способностью. Пользовательский процесс может добавлять сообщения в очередь сообщений или читать сообщения из очереди сообщений.

По сравнению с конвейерной связью очереди сообщений имеют преимущество, что для каждого сообщения указывается конкретный тип сообщения. При получении не требуется следовать порядку очереди, но можно получать сообщения определенных типов в соответствии с настраиваемыми условиями.

Система обмена сообщениями:

* **Метод прямой связи.** Процесс отправки использует примитивы отправки, предоставляемые ОС, для прямой отправки сообщения целевому процессу. В этом методе связь устанавливается между одной парой взаимодействующих процессов, и между каждой парой существует только одна связь.
* **Косвенный метод связи.** Процессы отправки и получения отправляют и получают сообщения через общие объекты (почтовые ящики).Непрямое взаимодействие устанавливается, только когда процессы совместно используют общий почтовый ящик. Каждая пара процессов совместно использует несколько каналов связи. Ссылка может общаться со многими процессами и может быть двунаправленной или однонаправленной.

1. **Общая память** (разделяемая**) –** это память, разделяемая между двумя или более процессами, которые устанавливаются с использованием общей памяти между всеми процессами. Этот тип памяти требует защиты друг от друга путем синхронизации доступа между всеми процессами.

Общая память предназначена для отображения раздела памяти, к которому могут получить доступ другие процессы. Эта часть общей памяти создается одним процессом, но несколько процессов могут получить к нему доступ, т.е. она позволяет двум или более процессам совместно использовать данную область хранения. Совместно используемая память - это самый быстрый метод IPC, специально разработанный с учетом низкой эффективности других методов межпроцессного взаимодействия. Он часто используется в сочетании с другими механизмами связи (такими как семафоры) для достижения синхронизации и связи между процессами.

Одним из основных преимуществ использования общей памяти является то, что процесс может напрямую читать и записывать в память данные. Для таких методов связи, как каналы и очереди сообщений, он должен копировать данные четыре раза: в ядре и в пространстве пользователя, в то время как общая память копирует только дважды (один раз из входного файла в общую область памяти, другой раз из общей памяти в выходной файл).

1. **Семафоры** (semaphore) - это счетчик, который можно использовать для управления доступом к общим ресурсам несколькими процессами. Используется для достижения взаимного исключения и, как средство синхронизации между процессами и между разными потоками в одном процессе.

Семафор - это целая переменная, значение которой можно опрашивать и менять только при помощи неделимых (атомарных) операций. Двоичный семафор может принимать только значения 0 или 1 (может принимать целые неотрицательные значения). В приложениях как правило требуется использование более одного семафора, ОС должна представлять возможность создавать множества семафоров.

Над каждым семафором, принадлежащим некоторому множеству, можно выполнить любую из трех операций:

* увеличить значение;
* уменьшить значение;
* дождаться обнуления.

1. **Сокеты** (socket)- это также механизм межпроцессного взаимодействия, который, в отличие от других механизмов может использоваться для связи между различными процессами.

Сокеты обеспечивают двухстороннюю связь типа «точка-точка» между двумя процессами. Они являются основными компонентами межсистемной и межпроцессной связи. Каждый сокет представляет собой конечную точку связи, с которой может быть совмещено некоторое имя. Он имеет определенный тип, один процесс или несколько, связанных с ним процессов.

**Основные типы сокетов:**

* Поточный - обеспечивает двухсторонний, последовательный, надежный, и недублированный поток данных без определенных границ.
* Датаграммный - поддерживает двухсторонний поток сообщений. Приложение, использующее такие сокеты, может получать сообщения в порядке, отличном от последовательности, в которой эти сообщения посылались.
* Сокет последовательных пакетов - обеспечивает двухсторонний, последовательный, надежный обмен датаграммами фиксированной максимальной длины.
* Простой сокет - обеспечивает доступ к основным протоколам связи.

1. **Сигналы (**sinal) - это более сложный метод связи, используемый для уведомления принимающего процесса о том, что произошло событие. Исходный процесс отправит сигнал, который распознается по номеру, а целевой процесс обработает его.

Сигналы являются программными прерываниями, которые посылаются процессу, когда случается некоторое событие. Сигналы могут возникать синхронно с ошибкой в приложении, например, SIGFPE (ошибка вычислений с плавающей запятой) и SIGSEGV (ошибка адресации), но большинство сигналов является асинхронными. Сигналы могут посылаться процессу, если система обнаруживает программное событие, например, когда пользователь дает команду прервать или остановить выполнение, или получен сигнал на завершение от другого процесса. Сигналы могут прийти непосредственно от ядра ОС, когда возникает сбой аппаратных средств ЭВМ. Система определяет набор сигналов, которые могут быть отправлены процессу. При этом каждый сигнал имеет целочисленное значение и приводит к строго определенным действиям.