8. Передача данных между процессами

8.1. Каналы передачи данных

Отправители и адресаты

- Под ***обменом данными между параллельными процессами*** понимается пересылка данных от одного потока к другому потоку, предполагая, что эти потоки выполняются в контексте разных процессов.
- Поток, который посылает данные другому потоку, называется ***отправителем***.
- Поток, который получает данные от другого потока, называется ***адресатом*** или ***получателем***.

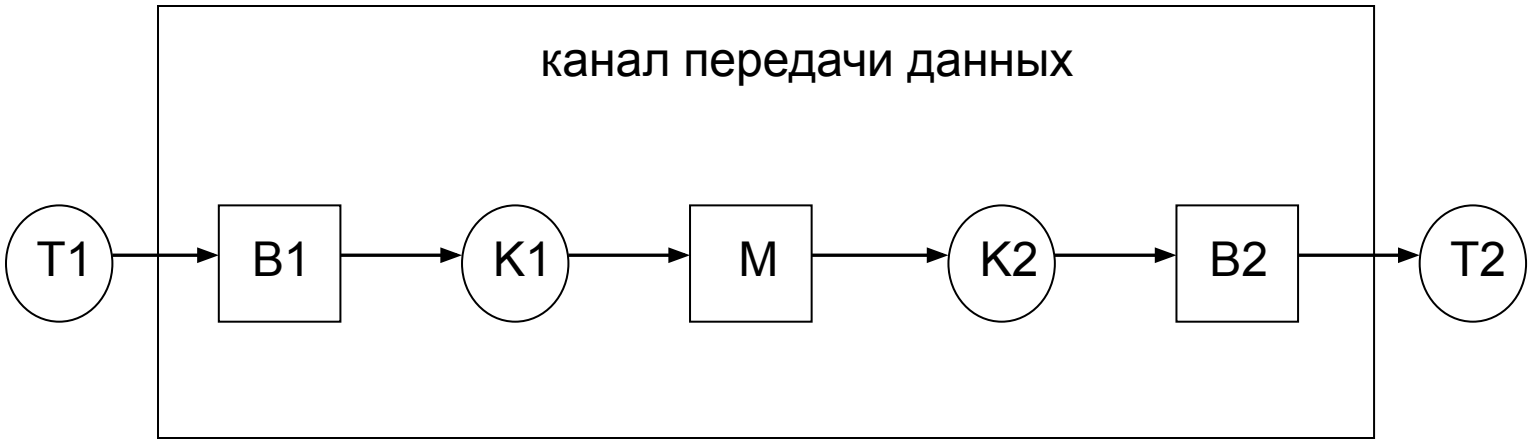
Обмен данными между потоками одного процесса

- Если потоки выполняются в контексте *одного процесса*, то обмен данными между ними можно организовать, используя глобальные переменные и средства синхронизации потоков.

Обмен данными между потоками разных процессов

- Если потоки выполняются в контекстах разных процессах, то потоки не могут обращаться к общим переменным.
- В этом случае для обмена данными между процессами создается ***канал передачи данных***, который является объектом ядра операционной системы и представляет собой область памяти, разделяемую несколькими процессами и используемую ими для обмена данными.

Схема канала передачи данных



T1, T2 – потоки пользователя

B1, B2 – буферы ввода-вывода

K1, K2 – потоки ядра операционной системы

M – общая память

Порядок работы канала передачи данных

- Пересылка данных из потока T1 в поток T2 происходит следующим образом:
  1. Пользовательский поток T1 записывает данные в буфер B1, используя специальную функцию ядра операционной системы;
  2. Поток K1 ядра операционной системы читает данные из буфера B1 и записывает их в общую память M;
  3. Поток K2 ядра операционной системы читает данные из общей памяти M и записывает их в буфер B2;
  4. Пользовательский поток T2 читает данные из буфера B2.

Реализация канала

- Обычно канал реализуется как кольцевой буфер, работающий по принципу ***FIFO***.
- Для работы с каналом могут использоваться такие же функции ввода-вывода, как и для работы с файлами.

Способы передачи данных по каналам

- Различают два способа передачи данных по каналам:
  - потоком;
  - сообщениями.
- Если данные передаются непрерывной последовательностью байтов, то такая пересылка данных называется ***передача данных потоком***.
- Если же данные пересылаются группами байтов, то такая группа байтов называется ***сообщением***, а сама пересылка данных называется ***передачей данных сообщениями***.

8.2. Связи между процессами

- Прежде чем пересылать данные между процессами, нужно установить между этими процессами связь.
- Связь между процессами устанавливается как на **физическом** (аппаратном), так и **логическом** (программном) уровнях.

Направления передачи данных

- С точки зрения направления передачи данных различают следующие виды связей:
  - ***полудуплексная связь*** – данные по этой связи могут передаваться только в одном направлении;
  - ***дуплексная связь*** – данные по этой связи могут передаваться в обоих направлениях.

Топологии связей

- Теперь, предполагая, что рассматриваются только полудуплексные связи, определим возможные топологии связей.
- Под ***топологией связи*** будем понимать конфигурацию связей между процессами отправителями и получателями.

Виды связей

- С точки зрения топологии различают следующие виды связей:

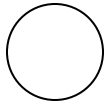
  $1 \rightarrow 1$ - между собой связаны только два процесса;

  $1 \rightarrow N$ - один процесс связан с N процессами;

  $N \rightarrow 1$ - каждый из N процессов связан с одним процессом;

  $N \rightarrow M$ - каждый из N процессов связан с каждым из M процессов.

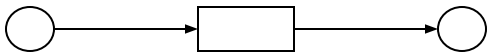
Обозначения

 - процесс

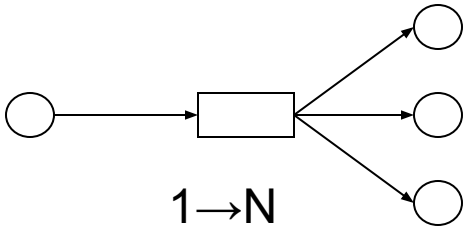
 - канал

 - связь

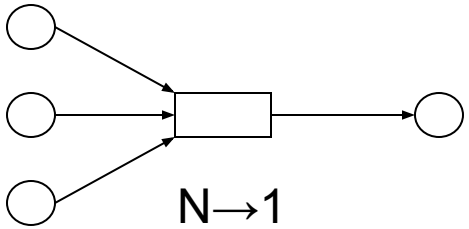
Топология связей между процессами



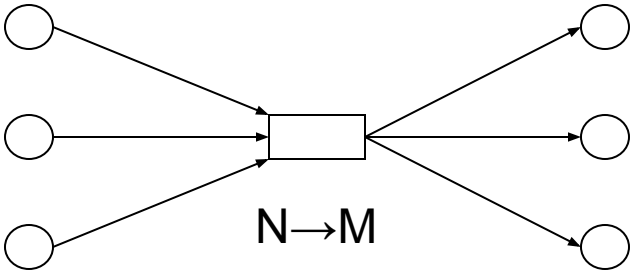
1→1



1→N



N→1



N→M

Порты и почтовые ящики

- Канал передачи данных, реализующий топологию N→1 обычно называется ***портом***.
- Канал передачи данных, реализующий топологию N→M обычно называется ***почтовым ящиком***.

Порядок разработки систем обмена данными

1. При разработке систем с обменом данными между процессами, прежде всего, должна быть выбрана топология связей и направления передачи данных по этим связям.
2. После этого в программах реализуются выбранные связи между процессами, используя функции операционной системы, предназначенные для установки связи между процессами.

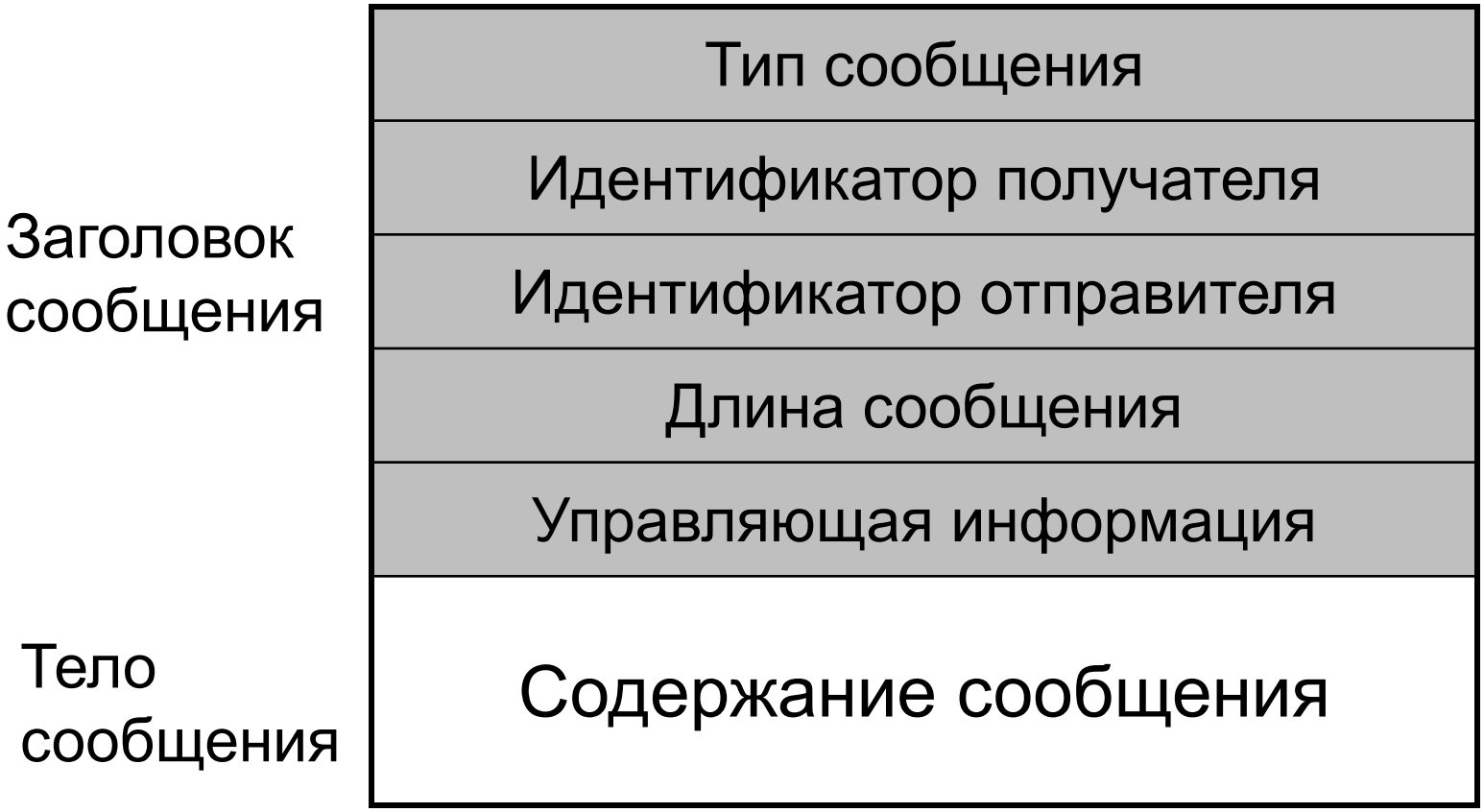
Функции для установки связей между процессами

- Для установки связей между процессами обычно используются функции типа:
  - **connect** – установить связь;
  - **disconnect** – разорвать связь.
- Эти функции, а также функции для обмена данными между процессами обеспечивает система передачи данных, которая обычно является частью ядра операционной системы.

8.3. Передача сообщений

- Обмен сообщениями между процессами выполняется при помощи двух функций:
  - ***send*** - послать сообщение;
  - ***receive*** - получить сообщение.

Структура сообщения



Типы адресации процессов

- При передаче сообщений может использоваться прямая или косвенная адресация процессов.

Прямая адресация процессов

- При ***прямой адресации*** процессов в функциях *send* и *receive* явно указываются процессы отправитель и получатель.
- В этом случае функции обмена данными имеют следующий вид:
  - *send(P, сообщение)* - послать сообщение процессу P;
  - *receive(Q, сообщение)* - получить сообщение от процесса Q.

Косвенная адресация процессов

- При ***косвенной адресации*** в функциях *send* и *receive* указываются не адреса процессов, а имя канала связи, по которому передается сообщение.
- В этом случае функции обмена данными имеют следующий вид:
  - *send(S, сообщение)* - послать *сообщение* по каналу связи S;
  - *receive(R, сообщение)* - получить *сообщение* по каналу связи R.
- В последнем случае сообщение могут получать все процессы, подключенные к каналу связи R.

Симметричная и асимметричная адресация

- Адресация процессов может быть симметричной и асимметричной.
- Если при передаче сообщений используется только прямая или только косвенная адресация, то такая адресация процессов называется ***симметричной***.
- В противном случае адресация процессов называется ***асимметричной***.

Адресация в системах клиент-сервер

- Асимметричная адресация процессов используется в системах клиент-сервер.
- В этом случае клиенты знают адрес сервера и посылают ему сообщения, используя функцию
  - *send(Server, сообщение)*
- А сервер «слушает» канал связи и принимает сообщения от всех клиентов, используя функцию
  - *receive(Connection, сообщение)*
- Часто эта функция так и называется listen (слушать).

Протокол

- Набор правил, по которым устанавливаются связи и передаются данные между процессами, называется ***протоколом***.

8.4. Синхронный и асинхронный обмен данными

- При передаче данных различают синхронный и асинхронный обмен данными.

Синхронное и асинхронное отправление сообщения

- Если поток отправитель, отправив сообщение функцией *send*, блокируется до получения этого сообщения потоком адресатом, то такое *отправление сообщения* называется ***синхронным***.
- В противном случае *отправление сообщения* называется ***асинхронным***.

Синхронное и асинхронное получение сообщения

- Если поток получатель, вызвавший функцию *receive*, блокируется до тех пор, пока не получит сообщение, то такое *получение сообщения* называется ***синхронным***.
- В противном случае *получение сообщения* называется ***асинхронным***.

Синхронный и асинхронный обмен сообщениями

- *Обмен сообщениями* называется ***синхронным***, если поток отправитель синхронно передает сообщения, а поток адресат синхронно принимает эти сообщения.
- В противном случае *обмен сообщениями* называется ***асинхронным***.
- Пересылка данных потоком всегда происходит синхронным образом, так как в этом случае между отправителем и получателем устанавливается непосредственная связь.

Рандеву

- Синхронный обмен данными в случае прямой адресации процессов называется ***рандеву*** (rendezvous), что переводится с французского языка как «встреча».
- Такой механизм обмена сообщениями используется в языке программирования Ада.

8.5. Буферизация

- ***Буфером*** называется *вместимость связи между процессами*, то есть количество сообщений, которые могут одновременно пересылаться по этой связи.

Типы буферизации

- Существенно различаются три типа буферизации:
  - *нулевая вместимость связи* (нет буфера), в этом случае возможен только синхронный обмен данными между процессами;
  - *ограниченная вместимость связи* (ограниченный буфер), в этом случае, если буфер полон, то отправитель сообщения должен ждать очистки буфера хотя бы от одного сообщения;
  - *неограниченная вместимость связи* (неограниченный буфер), в этом случае отправитель никогда не ждет при отправке сообщения.

- Как видно из этих определений типы буферизации тесно связаны с синхронизацией передачи данных и поэтому также должны учитываться при разработке систем, которые используют обмен данными между процессами.

8.6. Анонимные каналы в Windows

- ***Анонимным каналом*** называется объект ядра операционной системы, который обеспечивает передачу данных между процессами, выполняющимися на одном компьютере.
- Процесс, который создает анонимный канал, называется ***сервером*** *анонимного канала*.
- Процессы, которые связываются с анонимным каналом, называются ***клиентами*** *анонимного канала*.

Свойства анонимных каналов

- не имеют имени;
- полудуплексные;
- передача данных потоком;
- синхронный обмен данными;
- возможность моделирования любой топологии связей.

Порядок работы с анонимным каналом

- создание анонимного канала сервером;
- соединение клиентов с каналом;
- обмен данными по каналу;
- закрытие канала.

Соединение клиентов с анонимным каналом

- Так как анонимный канал не имеет имени, то доступ к такому каналу имеют только родительский процесс-сервер и дочерние процессы-клиенты этого канала.
- Чтобы процесс-клиент наследовал дескриптор анонимного канала, этот дескриптор должен быть наследуемым.
- Явная передача наследуемого дескриптора процессу-клиенту анонимного канала может выполняться одним из следующих способов:
  - через командную строку;
  - через поля *hStdInput*, *hStdOutput* и *hStdError* структуры *STARTUPINFO*;
  - посредством сообщения *WM_COPYDATA*;
  - через файл.

Функции для работы с анонимным каналом

- *CreatePipe* – создание анонимного канала;
- *WriteFile* – запись данных в анонимный канал;
- *ReadFile* – чтение данных из анонимного канала;

8.7. Именованные каналы в Windows

- ***Именованным каналом*** называется объект ядра операционной системы, который обеспечивает передачу данных между процессами, выполняющимися на компьютерах в одной локальной сети.
- Процесс, который создает именованный канал, называется ***сервером*** *именованного канала*.
- Процессы, которые связываются с именованным каналом, называются ***клиентами*** *именованного канала*.

Свойства именованных каналов

- имеют имя, которое используется клиентами для связи с именованным каналом;
- могут быть как полудуплексные, так и дуплексные;
- передача данных может осуществляться как потоком, так и сообщениями;
- обмен данными может быть как синхронным, так и асинхронным;
- возможность моделирования любой топологии связей.

Порядок работы с именованными каналами

- создание именованного канала сервером:
- соединение сервера с экземпляром именованного канала;
- соединение клиента с экземпляром именованного канала;
- обмен данными по именованному каналу;
- отсоединение сервера от экземпляра именованного канала;
- закрытие именованного канала клиентом и сервером.

Функции для соединения с именованным каналом

- *CreateNamedPipe* – создание именованного канала;
- *ConnectNamedPipe* – соединение сервера с клиентом именованного канала;
- *DisconnectNamedPipe* – отсоединение сервера от именованного канала;
- *WaitNamedPipe* – ожидание клиентом свободного экземпляра именованного канала;
- *CreateFile* – соединение клиента с именованным каналом;

Функции для передачи данных по именованному каналу

- *WriteFile* – запись даннх в именованный канал;
- *ReadFile* – чтение данных из именованного канала;
- *PeekNamedPipe* – копирование данных из именованного канала;
- *TransactNamedPipe* – обмен сообщениями по именованному каналу;

Функции для работы с состоянием и свойствами именованного канала

- *GetNamedPipeHandleState* – определение состояния именованного канала;
- *SetNamedPipeHandleState* – изменение состяния именованного канала;
- *GetNamedPipeInfo* – получить информацию об атрибутах именованного канала;

8.8. Почтовые ящики в Windows

- ***Почтовым ящиком*** называется объект ядра операционной системы, который обеспечивает передачу сообщений от процессов-клиентов к процессам-серверам, выполняющимся на компьютерах в пределах локальной сети.
- Процесс, который создает почтовый ящик, называется ***сервером*** *почтового ящика*.
- Процессы, которые связываются с почтовым ящиком, называются ***клиентами*** *почтового ящика*.

Свойства почтовых ящиков

- имеют имя, которое используется клиентами для связи с почтовыми ящиками;
- направление передачи данных от клиента к серверу;
- передача данных осуществляется сообщениями;
- обмен данными может быть как синхронным, так и асинхронным.

Передача сообщений почтовыми ящиками

- Хотя передача данных осуществляется только от клиента к серверу, один почтовый ящик может иметь несколько серверов.
- Это происходит в том случае, если несколько серверов создают почтовые ящики с одинаковыми именами.
- Тогда все сообщения, которые посылает клиент в такой почтовый ящик, будут получать все серверы этого почтового ящика.
- Таким образом, можно сказать, что почтовые ящики обеспечивают однонаправленную связь типа "многие ко многим".
- При этом доставка сообщения от клиента к серверам почтового ящика не подтверждается системой.

Порядок работы с почтовым ящиком

- создание почтового ящика сервером:
- соединение клиента с почтовым ящиком;
- обмен данными через почтовый ящик;
- закрытие почтового ящика клиентом и сервером.

Функции для работы с почтовыми ящиками

- *CreateMailslot* – создание почтового ящика;
- *CreateFile* – соединение клиента с почтовым ящиком;
- *WriteFile* – запись данных в почтовый ящик;
- *ReadFile* – чтение данных из почтового ящика;
- *GetMailslotInfo* – получение информации о свойствах почтового ящика;
- *SetMailslotInfo* – изменение сервером времени ожидания от клиента.