Операционные системы

Лекция № 7

Взаимодействие и обмен данными между процессам

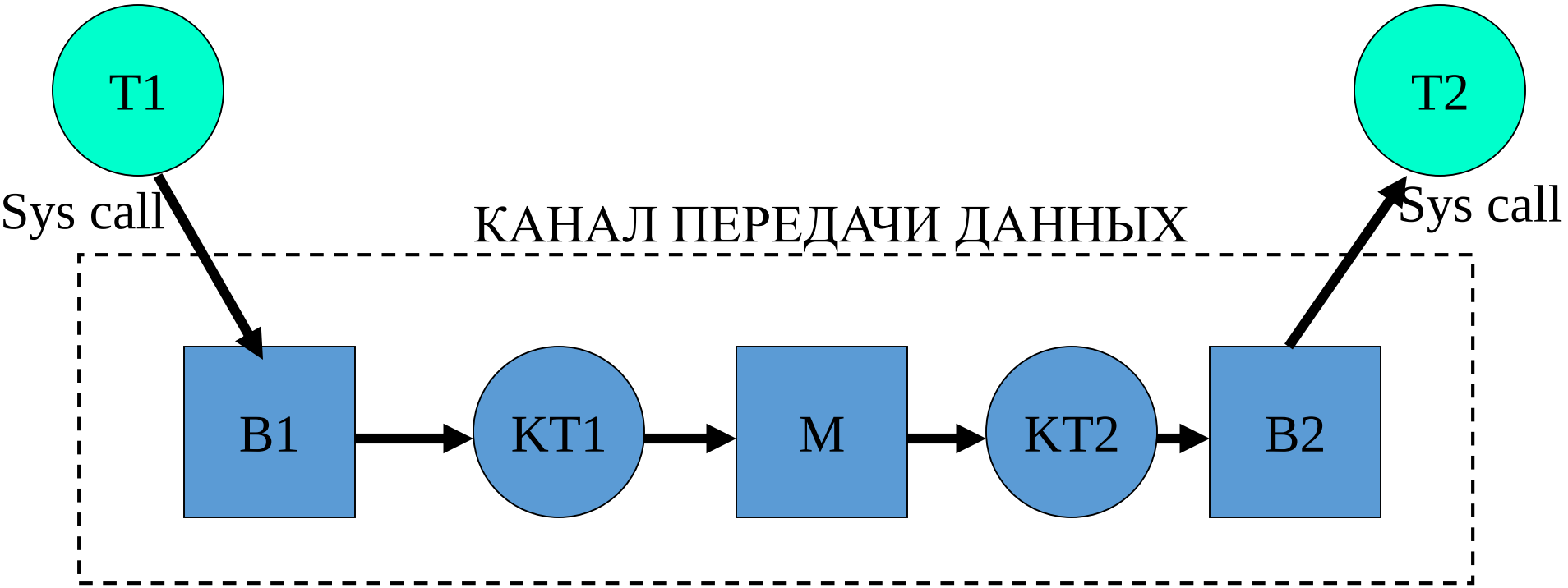
(4 часа)

Понятие межпроцессного взаимодействия.

- **Межпроцессное взаимодействие** (IPC, InterProcess Communication) – пересылка данных от одного потока другому потоку разных процессов.
- Поток, посылающий данные другому потоку называется **отправителем**.
- Поток, который принимает данные от другого потока – **адресатом** или **получателем**

Понятие межпроцессного взаимодействия.

Для обмена данными между процессами создается **канал передачи данных:**



Понятие межпроцессного взаимодействия.

Т1, Т2 – пользовательские потоки;

В1, В2 – входной и выходной буферы памяти;

КТ1, КТ2 – потоки ядра операционной системы;

М – некоторый механизм «общая память».

Понятие межпроцессного взаимодействия.

Способы передачи данных:

- **Потоком** – данные передаются непрерывной последовательностью байтов;
- **Сообщением** – данные передаются группами байтов - сообщениями.

Понятие межпроцессного взаимодействия.

Виды **межпроцессной связи**:

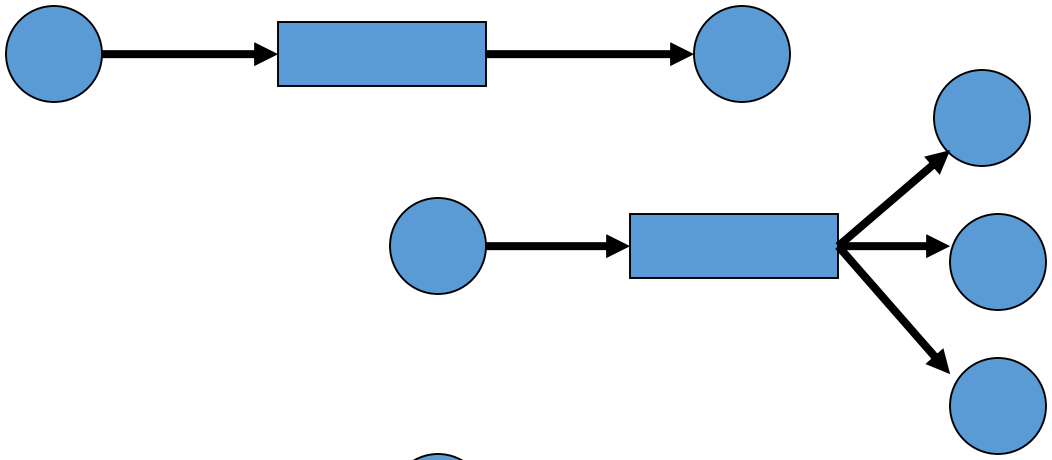
- **Полудуплексная связь** – данные по этой связи могут передаваться только в одном направлении;
- **Дуплексная связь** – данные могут передаваться в обоих направлениях.

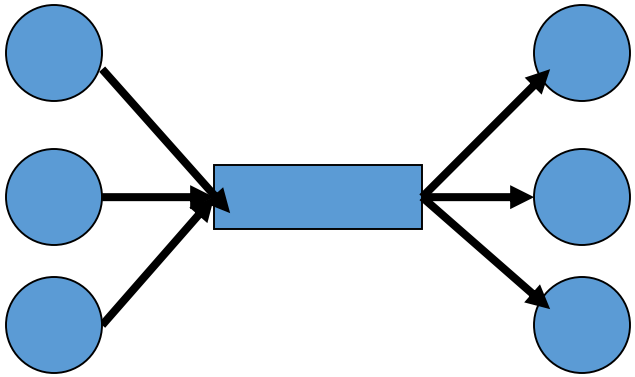
Топология связей – конфигурация связей между процессами-отправителями и адресатами.

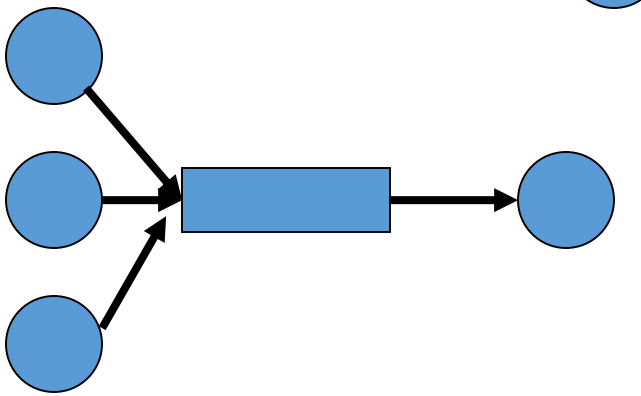
Понятие межпроцессного взаимодействия.

Топологии связей:

- 1 → 1;
- 1 → N;
- N → 1;
- N → M







Понятие межпроцессного взаимодействия.

Обмен сообщениями:

- `send` – послать сообщение;
- `receive` – получить сообщение.

Виды адресации:

- **Прямая** – явно указываются процессы отправитель и адресат;

```
send(process P1, m);
```

- **Косвенная** – указываются не адреса, а имя связи по которой передается сообщение.

```
send(connection Con, m);
```

Понятие межпроцессного взаимодействия.

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

При передаче данных различают:

- **Синхронный обмен данными** – поток обратившись к функции send ***блокируется*** до получения этого сообщения получателем. Иначе – **асинхронный обмен данными**.

Механизмы IPC в *nix.

Механизмы IPC *NIX:

1. Параметры exec-вызова;
2. Сигналы (только факт передачи и тип сигнала);
3. Сокеты (универсальное средство обмена данными – удаленный обмен и локальный);
4. Каналы (pipe);
5. FIFO (один из 7 типов файлов Linux);
6. Общая память (быстродействующий, требует дополнительной синхронизации, например, семафорами);
7. Общие файлы (например, проецированием одного и того же файла в память обоих процессов);
8. Очереди сообщений (System V и POSIX)

Общая память

Обмен данными реализуется через общий блок памяти – разделяемый сегмент, доступный потокам нескольких процессов.

Один из процессов выделяет блок памяти при помощи `shmget()`

Процессы подключают этот блок (получают его адрес) при помощи `shmat()`

Обмен данными сводится к чтению/записи из общего блока памяти и требует синхронизации доступа

После окончания обмена каждый процесс отключается от общей памяти – `shmdt()` и один из них удаляет его при помощи `shmctl()`

Общая память

Выделение разделяемого сегмента:

```
#include <sys/shm.h>
int shmget(
    key_t key,
    size_t size,
    int flags
);
```

Возвращает -1 в случае ошибки или идентификатор сегмента в случае успеха.

Общая память

Идентификатор сегмента – используется для доступа к сегменту процессом-создателем сегмента

Другие процессы получают доступ по ключу сегмента – `key` типа `key_t`:

- `IPC_PRIVATE` – если задается динамически функцией `shmget`;
- Генерируется функцией ftok() (file to key);
- Может задаваться статически заранее;

`size` – размер сегмента в байтах (с учётом выравнивания);

`flags` – права доступа к сегменту и вид поведения при создании

Общая память

`flags:`

`IPC_CREAT` – создать разделяемый сегмент памяти, если с таким ключом его нет;

`IPC_EXCL` – вызвать исключение `EEXIST` при попытке создать разделяемый сегмент с существующим ключом;

Права доступа в формате <владелец><группа><остальные> (каждое из значений получается заданием битов rwx с весами 4, 2 и 1 соответственно (или в S_I…)

Например, 0644

Общая память

Доступ из другого процесса:

- Вызов `shmget()` по известному ключу;
- Получение идентификатора разделяемого сегмента от процесса, его создавшего

Для работы с разделяемым сегментом процессы должны получить его адрес:

```
#include <sys/shm.h>
void* shmat(int id, void* address,
 int flags);
```

Общая память

`id` – идентификатор запрашиваемого сегмента;

`address` – адрес, с которого будет начинаться разделяемый сегмент в адресном пространстве процесса или `NULL` (адрес будет назначен `shmat`);

`flags` – флаги настройки, например SHM_RDONLY только для чтения (обычно 0).

Результат: адрес разделяемого сегмента или `(void*) -1` в случае ошибки

Общая память

По окончании использования разделяемого сегмента каждый процесс должен его удалить:

```
#include <sys/shm.h>
int shmdt (void* address);
```

Возвращает в случае успеха `0`, в случае ошибки `-1`.

`address` – указывает адрес размещения разделяемого сегмента (`shmat`)

Общая память

Управление разделяемым сегментом выполняется при помощи:

```
#include <sys/shm.h>
int shmctl(int id, int command, struct shmid_ds* desc);
```

Возвращает в случае успеха `0`, в случае ошибки `-1`.

`id` – идентификатор разделяемого сегмента над которым выполняется `command`;

`command` – команда, выполняемая над сегментом;

`desc` – указать на структуру, в которую заносятся данные о сегменте

Общая память

command:

`IPC_STAT` – получить информацию о сегменте в структуру по адресу `desc`;

`IPC_RMID` – удалить сегмент с идентификатором `id`.

Общая память

Управлять разделяемыми сегментами можно из консоли Linux:

ipcs – просмотр средств IPC (в том числе разделяемых сегментов);

ipcrm – удаление средства IPC (может быть выполнено владельцем разделяемого сегмента)

FIFO

FIFO – аналог именованных каналов в Windows, реализующий доступ по принципу First In First Out

Представлены специальными файлами в файловой системе

Позволяют использовать стандартные вызовы ФС:

`open()`

`read()`

`write()`

`close()`

FIFO

Создание FIFO:

```
#include <sys/stat.h>
int mkfifo (const char* pathname,
    mode_t mode);
```

Возвращает в случае успеха `0`, в случае ошибки `-1`.

`pathname` – имя создаваемого файла FIFO;

`mode` – права доступа к файлу FIFO в формате `S_I[R,W,X][USR,GRP,OTH]`

Например, `S_IRUSR`

FIFO

После создания FIFO процессы открывают канал на чтение/запись используя вызов open() с флагом O_RDONLY – для читающей стороны или O_WRONLY – для пишущей.

При этом, читающая сторона блокируется пока не подключится пишущая и наоборот.

FIFO

FIFO реализуют байтовый поток, следовательно нужно применять одну из 3 техник:

- Использование символа разделителя (например, \n);
- Использование заголовка с указанием длины сообщения;
- Передача порциями фиксированной длины

FIFO

Удаление FIFO:

```
#include <unistd.h>
int unlink (const char* filename);
```

Возвращает в случае успеха `0`, в случае ошибки `-1`.

filename – имя файла для удаления

Механизмы IPC в *nix.

Подробнее:

Иванов Н.Н. – Программирование в Linux. Самоучитель. – часть 5;

Теренс Чен – Системное программирование на C++ для UNIX. – с.333

Механизмы IPC Windows

1. Сообщение WM_COPYDATA;
2. Анонимные каналы (anonymous channels);
3. Именованные каналы (named pipes);
4. Почтовые ящики (mailslots);
5. Файлы;
6. Сокеты (sockets);
7. Разделяемая память (shared memory);
8. Буфер обмена (clipboard);
9. Динамический обмен данными (DDE);
10. COM/DCOM;
11. Microsoft Message Queue (MSMQ);
12. Удаленный вызов процедур (Remote Procedure Call, RPC).

Механизмы IPC Windows
Анонимные каналы

Анонимные каналы – объекты ядра ОС, обеспечивающие передачу данных между процессами, выполняющимися на одном компьютере.

Основные характеристики:

- Не имеют имени;
- Полудуплексные;
- Передача данных потоком;
- Синхронный обмен данными;
- Возможность моделирования любой топологии связей.

Механизмы IPC Windows Анонимные каналы

Алгоритм работы с анонимным каналом:

- Создание анонимного канала сервером;
- Соединение клиентов с каналом;
- Обмен данными по каналу;
- Закрытие канала.

Механизмы IPC Windows
Анонимные каналы

Создание анонимного канала сервером

```
BOOL CreatePipe(
  PHANDLE hReadPipe,
  PHANDLE hWritePipe,
  LPSECURITY_ATTRIBUTES
   lpPipeAttributes,
  DWORD nSize
);
```

Механизмы IPC Windows
Анонимные каналы

Здесь,

`PHANDLE hReadPipe` – дескриптор для чтения из канала;

`PHANDLE hWritePipe` – дескриптор для записи в канал;

`LPSECURITY_ATTRIBUTES lpPipeAttributes` – атрибуты защиты;

`DWORD nSize` – размер буфера в байтах (=0 – значение по умолчанию, - выбирается ОС).

В случае удачного завершения возвращает ненулевое значение, иначе – `FALSE`.

Механизмы IPC Windows
Анонимные каналы

Соединение клиентов с каналом

- Каналы не имеют имени – для соединения следует передать один из дескрипторов;
- Дескриптор должен быть наследуемым;
- Процесс-клиент должен быть дочерним.

Механизмы IPC Windows
Анонимные каналы

Передача дескриптора:

- Через командную строку;
- Через поля `hStdInput`, `hStdOutput`, `hStdError` структуры `STARTUPINFO`;
- Посредством сообщения `WM_COPYDATA`;
- Через файл.

Механизмы IPC Windows
Анонимные каналы

Обмен данными по каналу

- Используются те же функции, что и для работы с файлом: WriteFile() и ReadFile():

```
BOOL WriteFile(
  HANDLE hAnonymousPipe,
  LPCVOID lpBuffer,
  DWORD nNumberOfBytesToWrite,
  LPDWORD lpNumberOfBytesWritten,
  LPOVERLAPPED lpOverlapped      // NULL!
);
```

При успешном завершении – ненулевое значение, иначе FALSE

Механизмы IPC Windows
Анонимные каналы

```
BOOL ReadFile(
  HANDLE hAnonymousPipe,
  LPVOID lpBuffer,
  DWORD nNumberOfBytesToRead,
  LPDWORD lpNumberOfBytesRead,
  LPOVERLAPPED lpOverlapped   // NULL !
);
```

При успешном завершении – ненулевое значение,
иначе FALSE

Механизмы IPC Windows
Анонимные каналы

Закрытие канала

После завершения обмена данными процессы **должны закрыть** дескрипторы записи и чтения анонимного канала (**CloseHandle**).

Механизмы IPC Windows
Анонимные каналы

```
HANDLE hWPipe, hRPipe;
SECURITY_ATTRIBUTES sa;

sa.nLength = sizeof(sa);
sa.lpSecurityDescriptor = NULL;
sa.bInheritHandle = TRUE;

if(! CreatePipe(&hWPipe,&hRPipe,&sa,NULL)
{
  cerr << "Error " << GetLastError() << endl;
  return 1;
}
… // работа с анонимным каналом
```

Механизмы IPC Windows
Анонимные каналы

```
wsprintf(comline, "C:\\child.exe %d",
    hWPipe);
…
// запускаем процесс и передаем ему
// наследуемые дескрипторы
…
CloseHandle(hWPipe);
… // читаем данные из канала
char Message[128]; DWORD Readed, Data=128;
if (! ReadFile(hRPipe, Message, Data,
    &Readed, NULL))
{ … // Обработка ошибки … }
cout << "Получено сообщение:\t" << Message;
```

Механизмы IPC Windows
Анонимные каналы

Анонимные каналы позволяют **перенаправить** стандартный I/O:

- При создании консольного процесса стандартные файлы и стандартные потоки I/O связываются с дескрипторами, заданными в полях `hStdInput`, `hStdOutput`, `hStdError` структуры `STARTUPINFO`.
- Если в эти поля записать дескрипторы анонимного канала, то для передачи данных можно использовать функции стандартного I/O.

Такая процедура называется ***перенаправлением стандартного ввода-вывода***.

Механизмы IPC Windows
Анонимные каналы

Для этого:

1. Создать наследуемый дескриптор анонимного канала;
2. Добавить в поле dwFlags структуры STARTUPINFO для дочернего процесса флаг STARTF_USESTDHANDLES;
3. Установить нужные из полей hStdInput, hStdOutput, hStdError;
4. Запустить дочерний процесс с флагом наследования дескрипторов.

Механизмы IPC Windows
Именованные каналы

Именованный канал – объект ядра ОС, который обеспечивает передачу данных между процессами, выполняющимися в пределах одной локальной сети.

Характеристики:

- Имеют имя, используемое клиентами для связи;
- М.б. как полудуплексные, так и дуплексные;
- Передача данных как потоком, так и сообщениями;
- Обмен данными как синхронный, так и асинхронный;
- Возможно моделирование любой топологии связей.

Механизмы IPC Windows
Именованные каналы

Алгоритм работы с именованными каналами:

1. Создание именованного канала сервером;
2. Соединение сервера с экземпляром именованного канала;
3. Соединение клиента с экземпляром именованного канала;
4. Обмен данными по именованному каналу;
5. Отсоединение сервера от экземпляра именованного канала;
6. Закрытие именованного канала клиентом и сервером (CloseHandle()).

Механизмы IPC Windows
Именованные каналы

Создание именованного канала сервером:

```
HANDLE CreateNamedPipe(
  LPCTSTR lpName,
  DWORD dwOpenMode,
  DWORD dwPipeMode,
  DWORD nMaxInstances,
  DWORD nOutBufferSize,
  DWORD nInBufferSize,
  DWORD nDefaultTimeOut,
  LPSECURITY_ATTRIBUTES lpSecurityAttributes
);
```

Механизмы IPC Windows
Именованные каналы

В случае удачи возвращает дескриптор именованного канала, иначе:

- `INVALID_HANDLE_VALUE`;
- `ERROR_INVALID_PARAMETER` (если параметр `nMaxInstances > PIPE_UNLIMITED_INSTANCES`)

Параметры:

`lpName` – имя канала. Строка вида: "**`\\.\pipe\`**`PipeName`";

`dwOpenMode` – атрибуты канала

- `PIPE_ACCESS_DUPLEX`;
- `PIPE_ACCESS_INBOUND` – клиент пишет, сервер читает;
- `PIPE_ACCESS_OUTBOUND` – наоборот.

Механизмы IPC Windows
Именованные каналы

dwPipeMode – режим передачи данных

- PIPE_TYPE_BYTE – запись данных потоком;
- PIPE_TYPE_MESSAGE – запись данных сообщениями;
- PIPE_READMODE_BYTE – чтение потоком;
- PIPE_READMODE_MESSAGE – чтение сообщениями

(по умолчанию – потоком)

- PIPE_WAIT – синхронный обмен данными;
- PIPE_NOWAIT – асинхронный обмен;

nMaxInstances – максимальное количество экземпляров канала (от 1 до PIPE_UNLIMITED_INSTANCES (255));

Механизмы IPC Windows
Именованные каналы

`nOutBufferSize` – размер выходного буфера;
`nInBufferSize` – размер входного буфера;
`nDefaultTimeOut` – время ожидания связи с сервером для клиента;
`lpSecurityAttributes` – атрибуты безопасности.

Механизмы IPC Windows
Именованные каналы

Соединение сервера с экземпляром именованного канала:

```
BOOL ConnectNamedPipe(
  HANDLE hNamedPipe,
  LPOVERLAPPED lpOverlapped
);
```

В случае успеха возвращает ненулевое значение, иначе FALSE.

По завершению обмена сервер должен отсоединиться:

```
BOOL DisconnectNamedPipe(
  HANDLE hNamedPipe);
```

Механизмы IPC Windows
Именованные каналы

Соединение клиента с экземпляром именованного канала:

Выполняется в 2 шага:

1. Определение наличия доступного канала для подключения к серверу с помощью `WaitNamedPipe();`
2. Установление связи с этим каналом с помощью `CreateFile()`.

Механизмы IPC Windows
Именованные каналы

```
BOOL WaitNamedPipe(
  LPCTSTR lpNamedPipeName,
  DWORD nTimeOut
);
```

lpNamedPipeName – имя канала в виде:
"\\servername\pipe\pipename"

nTimeOut – время ожидания клиентом подключения к серверу:

- Интервал в миллисекундах;
- NMPWAIT_USE_DEFAULT_WAIT – определяется параметром сервера nDefaultTimeOut;
- NMPWAIT_WAIT_FOREVER – бесконечно.

Механизмы IPC Windows
Именованные каналы

```
HANDLE CreateFile(
LPCTSTR lpNamedPipeName,
DWORD dwDesiredAccess,
DWORD dwShareMode,
LPSECURITY_ATTRIBUTES
   lpSecurityAttributes,
OPEN_EXISTING,
0,
NULL);
```

Механизмы IPC Windows
Именованные каналы

Обмен данными по именованному каналу:

- `ReadFile();`
- `WriteFile();`

(доступен асинхронный режим обмена данными).

Максимальный объем данных для записи одной операцией: 65535 байт.

Механизмы IPC Windows
Именованные каналы

```
HANDLE hNamedPipe;
hNamedPipe =
    CreateNamedPipe("\\\\.\\pipe\\demo",
    PIPE_ACCESS_INBOUND, PIPE_TYPE_MESSAGE |
    PIPE_WAIT, 1, 512, 512, INFINITE, NULL);
if(hNamedPipe = = INVALIDE_HANDLE_VALUE)
{
  // обработка ошибки
}
…
```

Механизмы IPC Windows
Именованные каналы

```
if(!ConnectNamedPipe(hNamedPipe,NULL))
{
  CloseHandle(hNamedPipe);
// обработка ошибки
}
char Message[128]; DWORD Readed, mSize = 128;
if (! ReadFile(hNamedPipe, Message, mSize,
    &Readed, NULL)
{
// обработка ошибки
}
// работа с полученными данными
```

Механизмы IPC Windows
Именованные каналы

```
HANDLE hNamedPipe;
char Npname[] = "\\\\teststation\\pipe\\demo";
hNamedPipe = CreateFile(Npname, GENERIC_WRITE,
    FILE_SHARE_READ, NULL, OPEN_EXISTING, 0,
    NULL);
if (hNamedPipe = = INVALID_HANDLE_VALUE)
{
// обработка ошибки
}
// работа с именованным каналом
CloseHandle(hNamedPipe);
```

Механизмы IPC Windows
Именованные каналы

Другие функции для работы с именованными каналами:

`PeekNamedPipe()` – копирует данные из именованного канала в буфер, не удаляя данные из канала;

`TransactNamedPipe()` – передает транзакцию по именованному каналу (одновременная операция чтения и записи из/в именованный канал);

`GetNamedPipeHandleState()` – получает информацию о состоянии именованного канала.

`SetNamedPipeHandleState()` – изменяет характеристики именованного канала.

`GetNamedPipeInfo()` – получает информацию об NamedPipe.

Механизмы IPC Windows
Почтовые ящики

Почтовый ящик (mailslot) – объект ядра ОС, который обеспечивает передачу сообщений от процессов-клиентов к процессам-серверам в пределах локальной сети.

Характеристики:

- Имеет имя;
- Направление передачи данных – от клиента к серверу;
- Передача данных сообщениями;
- Обмен данными синхронный и асинхронный.

Механизмы IPC Windows
Почтовые ящики

Алгоритм работы:

1. Создание почтового ящика сервером.
2. Соединение клиента с почтовым ящиком.
3. Обмен данными через почтовый ящик.
4. Закрытие почтового ящика клиентом и сервером.

Механизмы IPC Windows
Почтовые ящики

Создание почтового ящика сервером.

```
HANDLE CreateMailslot(
  LPCTSTR lpName,
  DWORD nMaxMessageSize,
  DWORD lReadTimeout,
  LPSECURITY_ATTRIBUTES lpSecurityAttributes
);
```

В случае успешного завершения возвращает дескриптор почтового ящика, иначе – INVALID_HANDLE_VALUE.

Механизмы IPC Windows
Почтовые ящики

`lpName` – имя mailslot в формате:
`\\.\mailslot\mailslotname`
`nMaxMessageSize` – максимальная длина сообщения в байтах;
`lReadTimeout` – время в миллисекундах в течение которого ReadFile ждет поступления данных в почтовый ящик (0 – MAILSLOT_WAIT_FOREVER);
`lpSecurityAttributes` – атрибуты безопасности.

Механизмы IPC Windows
Почтовые ящики

Соединение клиента с почтовым ящиком.

```
HANDLE CreateFile(
LPCTSTR lpMailslotName,
GENERIC_WRITE,
DWORD dwShareMode,
LPSECURITY_ATTRIBUTES lpsa,
OPEN_EXISTING,
0,
NULL
);
```

Механизмы IPC Windows
Почтовые ящики

- Почтовый ящик на локальном компьютере: `\\.\mailslot\имя`
- Почтовый ящик на указанном компьютере: `\\computename\mailslot\имя`
- Почтовый ящик в указанном домене: `\\domain\mailslot\имя`
- Почтовый ящик в текущем домене: `\\*\mailslot\имя`

Механизмы IPC Windows
Почтовые ящики

Обмен данными через почтовый ящик:
`WriteFile` (сторона клиента)
`ReadFile` (сторона сервера)

Закрытие почтового ящика клиентом и сервером:
`CloseHandle()`

Механизмы IPC Windows
Почтовые ящики

Получение информации о mailslot:

```
BOOL GetMailslotInfo(
    HANDLE hMailslot,
    LPDWORD lpMaxMessageSize,
    LPDWORD lpNextSize,
    LPDWORD lpMessageCount,
    LPDWORD lpReadTimeout
);
```

Механизмы IPC Windows
Почтовые ящики

hMailslot – дескриптор почтового ящика;
lpMaxMessageSize - max длина сообщения;
lpNextSize - длина следующего сообщения в почтовом ящике или MAILSLOT_NO_MESSAGE;
lpMessageCount - количество сообщений в почтовом ящике;
lpReadTimeout – период времени в течение которого ReadFile будет ожидать сообщение

Механизмы IPC Windows
Почтовые ящики

```
HANDLE hMailslot;

hMailslot =
    CreateMailslot("\\\\.\\mailslot\\demo", 0,
    MAILSLOT_WAIT_FOREVER, NULL);

if (hMailslot == INVALID_HANDLE_VALUE)
{
// обработка ошибок
}
```

Механизмы IPC Windows
Почтовые ящики

```
if(!GetMailslotInfo(hMailslot, NULL,
    &dwNextMessageSize,
    &dwMessageCount,NULL))
{// обработка ошибок}
while(dwMessageCount != 0)
{
 DWORD Readed;
 char* Message = new char[dwNextMessage];
 // читаем сообщение из mailslot
}
```