Лабораторная работа №7
Передача данных между процессами. Часть I
(2 часа)

Содержание: передача данных между процессами с использованием анонимных и именованных каналов, мэйлслотов (почтовых ящиков).

Цель: изучить основные способы использования объектов ядра анонимный канал (anonymous pipe), именованный канал (named pipe) и почтовый ящик (mailslot) для межпроцессного обмена в Windows.

Под передачей данных между процессами обычно понимается задача пересылки данных от одного потока другому потоку, предполагая, что эти потоки выполняются в контекстах различных процессов. В таком случае подобный обмен представляет определенную проблему, так как у таких потоков нет глобальных переменных, доступных им одновременно.

В операционных системах семейства Windows существует целый арсенал средств для реализации подобного обмена, который также часто называют IPC (InterProcess Communication – межпроцессное взаимодействие).

Анонимные каналы. Анонимные каналы (anonymous pipes) это объекты ядра, обеспечивающие полудуплексную передачу данных между процессами, которые выполняются на одном и том же компьютере. Это позволяет использовать их для организации конвейера – перенаправления выхода одной программы на вход другой. Передача дескрипторов анонимного канала чаще всего осуществляется путём наследования от родительского процесса дочернему. Отсутствие именования несколько сужает сферу их применения.

Задание 1. Разработайте приложение для вывода списка файлов в указанном пользователем каталоге. Для получения списка файлов используйте анонимные каналы.

Почтовые ящики (mailslots) это объекты ядра, которые реализуют простую одностороннюю рассылку информации, основанную на передаче сообщений - дейтаграмм. С каждым ящиком может быть связано несколько записывающих процессов и несколько считывающих. Записывающему процессу неизвестно сколько клиентов получило сообщение (доставка не гарантируется), причем клиенты могут находиться не только на локальном компьютере, но и в любом месте сети. Размер сообщений ограничен.

Алгоритм использования почтовых ящиков:

1. Сервер создает дескриптор почтового ящика с заданным именем с помощью вызова `CreateMailslot`. Создать почтовый ящик можно только на локальном компьютере (формат имени смотрите ниже).
2. Сервер ожидает получения сообщения с помощью `ReadFile`.

3. Клиент (доступна только запись!) открывает почтовый ящик вызовом `CreateFile` и пишет сообщение с помощью `WriteFile`. В случае отсутствия на другой стороне сервера открытие ящика клиентом вызовет ошибку. Сообщение клиента может быть прочитано ВСЕМИ серверами.

4. По завершению обмена данными программы закрывают дескрипторы почтового ящика.

```
HANDLE CreateMailslot (LPCTSTR lpName, DWORD cdMaxMsg,
DWORD dwReadTimeout, LPSECURITY_ATTRIBUTES lpsa);
```

где `lpName` задает имя почтового ящика,

`cdMaxMsg` максимальный размер одного сообщения (0 – любой размер),

`dwReadTimeout` - тайм-аут ожидания сообщений в миллисекундах (0 – немедленный возврат, значение `MAILSLOT_WAIT_FOREVER` соответствует неограниченному времени ожидания) – на какой срок блокируется вызов `ReadFile`, если нет данных,

`lpsa` атрибуты безопасности. Вызов возвращает дескриптор почтового ящика, который следует использовать при обращении к `ReadFile` для чтения сообщений из почтового ящика.

Имя почтового ящика имеет следующий формат:

`\\.\mailslot\[path]mailslotname`

Имя должно быть уникальным в пределах локального компьютера (но не сети). Здесь «.» – указывает, что почтовый ящик создается на локальном компьютере.

При открытии ящика имя может быть указано в следующем виде:

`\\.\mailslot\[path]mailslotname`

для открытия почтового ящика на локальном компьютере, или

`\\computername\mailslot\[path]mailslotname`

для открытия почтового ящика на удаленном компьютере, или

`\\domainname\mailslot\[path]mailslotname`

для открытия ящиков в пределах заданного домена. Кроме того, при открытии сервера клиент может использовать имя

`\\*\mailslot\mailslotname`

, что приведет к обнаружению любого сервера в пределах главного домена.

Для двух последних случаев максимальный размер сообщения составляет 424 байта.

При открытии почтового ящика в `CreateFile` необходимо использовать параметры `FILE_SHARE_READ` и `OPEN_EXISTING`.

Именованные каналы (named pipes) являются дуплексными и могут использоваться не только для передачи информации между процессами на

локальном компьютере, но и для обмена информацией по сети. Именованные каналы предоставляют возможность двухстороннего обмена данными по одному каналу между различными процессами.

Для создания используется функция:

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

при этом имя канала задается в формате:

`\\.\pipe\[path]pipename`, где «`.`» означает локальный компьютер (создать канал на удаленном компьютере не возможно).

В случае ошибки функция возвращает `INVALID_HANDLE_VALUE`.

Для подключения к созданному каналу используется функция `CreateFile`, для которой в качестве имени файла передается строка вида:

`\\.\pipe\[path]pipename`, если открывается канал на локальном компьютере или `\\servername\pipe\[path]pipename` - если на другом компьютере.

Для определения подключения к именованному каналу клиента сервер использует функцию:

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

После окончания работы с клиентом сервер должен вызвать

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

для освобождения дескриптора канала. После этого сервер может продолжить работу с другим клиентом.

Для определения возможности подключения к серверу клиент использует функцию

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

которая возвращает состояние канала на сервере – может ли он принять вызов клиента (т.е. есть или нет незавершенный вызов `ConnectNamedPipe`).

Как и в случае почтовых ящиков, обмен данными по именованному каналу выполняется при помощи функции `ReadFile` и `WriteFile`.

Если клиент и сервер именованных каналов работают на разных компьютерах локальной сети и при создании именованного канала используются атрибуты безопасности по умолчанию (это означает, что он принадлежит пользователю, создавшему этот канал), то вход на компьютер с сервером и на компьютер с клиентом должен быть выполнен под одним и

тем же логином. Чтобы снять это ограничение можно использовать следующий код:

```
…
// объявляем дескриптор именованного канала,
// атрибуты защиты и
// дескриптор безопасности
HANDLE hNamedPipe;
SECURITY_ATTRIBUTES sa;
SECURITY_DESCRIPTOR sd;

…
// заполняем атрибуты безопасности,
// отключаем наследование
// дескрипторов
sa.bInheritHandle = FALSE;
sa.nLength = sizeof(sa);
// инициализируем дескриптор безопасности
InitializeSecurityDescriptor                    (&sd,
SECURITY_DESCRIPTOR_REVISION);
// Устанавливаем нужные права: DACL отсутствует
// Значит, параметры безопасности не устанавливаются
// и доступ разрешен всем
SetSecurityDescriptorDacl(&sd, TRUE, NULL, FALSE);
// созданный дескриптор используем в SECURITY_ATTRIBUTES
sa.lpSecurityDescriptor = &sd;

// создаем именованный канал с разрешением на доступ
// для всех
hNamedPipe                                          =
CreateNamedPipe("\\\\.\\pipe\\SupaDupaProtectionPipe",
PIPE_ACCESS_DUPLEX, PIPE_TYPE_MESSAGE | PIPE_WAIT, 255, 0,
0, INFINITE, &sa);

if (hNamedPipe = = INVALIDE_HANDLE_VALUE)
{
  cerr << "Ошибка создания именованного канала!" <<
endl;
  cerr << "Код ошибки: " << GetLastError() << endl;
  cin.get();
  return 0;
}
…
```

На стороне клиента изменения в код вносить не нужно.

После окончания работы с каналом, не забудьте закрыть его дескрипторы!

Задание 2. С использованием средств IPC операционной системы Windows (именованные каналы или почтовые ящики) разработайте клиент-серверное приложение для обмена короткими текстовыми сообщениями в пределах локальной сети. Приложение может быть реализовано как в виде двух отдельных программ – клиентской и серверной, так и в виде единого модуля.