2.Григолюк Э.И., Ковалев Ю.Д., Фильштинский Л.А. Изгиб слоя, ослабленного сквозными туннельными разрезами // ДАН СССР. - 1991, Том 317, Т1, с. 51-53.
3. Мусхелишвили Н.И. Сингулярные интегральные уравнения. - М.: Физматгиз, 1962, 511с.
4. Лурье А.И. К теории толстых плит // ПММ / - 1942, т.6, вып. 2/3, с. 151-168.
5. Белоцерковский С.М., Лифанов И.К. Численные методы в сингулярных интегральных уравнениях. - М.: Наука, 1985, 253с.

Поступила в редколлегию 29 декабря 1994г.

УДК 681.3.06

ПРОГРАММИРОВАНИЕ В ЛОКАЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СЕТЯХ С ИСПОЛЬЗОВАНИЕМ ПРОТОКОЛА IPX

Бирюков А.В., нач. отд., Любчак В.А., доц.

В статье освещаются некоторые вопросы, связанные с программированием в локальных вычислительных сетях, а именно: использование низкоуровневых протоколов передачи данных для решения проблем взаимодействия рабочих станций локальной сети.

В настоящее время разработано и активно используется большое количество различных принципов и методов построения локальных сетей, отличающихся как своей топологией, так и способом доступа к среде передачи [1]. Все они тесным образом связаны с так называемой эталонной моделью взаимодействия открытых систем (OSI), разработанной международной организацией по стандартизации (ISO). Предметом статьи является протокол передачи данных сетевого, транспортного и сеансового уровней эталонной модели OSI [1,2]. Именно на этих уровнях происходит подготовка, обработка, буферизация, отправка и получение пакетов данных в сети. Протоколы передачи данных в сети и выполняют вышеперечисленные функции. Для протоколов передачи данных международным институтом инженеров по электротехнике и радиоэлектронике были разработаны стандарты IEEE802 [1,2]. В дальнейшем мы остановимся на рассмотрении принципов программирования с использованием протокола IPX (InterNetwork Packet Exchange) - протокола межсетевой передачи пакетов, базирующегося на методе доступа к сети Ethernet (стандарт IEEE802.3). Известно, что наиболее быстрым, экономичным, с точки зрения использования оперативной памяти, является именно этот протокол [2].

Авторами был накоплен большой опыт по разработке различных сетевых приложений, ориентированных на применение функций протокола IPX. Рассмотрим несколько подробнее общую схему использования данного протокола. Для организации взаимодействия рабочих станций в сети с использованием протокола IPX необходимыми и достаточными условиями являются:

- наличие как минимум двух рабочих станций, объединенных между собой сетевыми коммуникациями на базе сетевых адаптеров Ethernet;
- наличие в оперативной памяти рабочих станций помимо операционной системы (MS-DOS, PS-DOS, Windows 3.x, OS/2) специальных резидентных программ (для MS-DOS, PC-DOS это модули IPX.COM или IPXODI.COM);
- разработанные с использованием драйвера IPX прикладные программы, которые собственно и управляют процессом взаимодействия рабочих станций.

Первое, что должна сделать прикладная программа, использующая протокол IPX, - убедиться, установлен ли драйвер протокола IPX. Затем

необходимо определить адрес вызова этого драйвера - точку входа API (Aplication Program Interface).

Существует два способа вызова IPX:

1. Необходимо установить регистры процессора, так как определено для каждой из функций IPX. Затем вызывается IPX, используя указатель функции, полученный во время теста на присутствие в оперативной памяти IPX.

2. Использование прерывания 7ah для вызова IPX. Это прерывание является второй точкой входа в IPX.

Рассмотрим фрагмент программы, написанный на языке программирования С, которая проводит тест на наличие в памяти компьютера драйвера IPX и определяет точку входа в IPX. Но прежде сделаем ряд принципиальных замечаний.

С развитием сетевых технологий сложились и активно используются различные подходы и способы написания программ, работающих в сети. Это можно сказать и о низкоуровневых программах, работающих непосредственно с протоколами передачи данных. Существует целый ряд инструментальных средств и библиотек функций, осуществляющих доступ к функциям драйверов протоколов передачи: инструментальные средства разработки сетевых приложений, библиотека функций Netware C Interface и инструментальное средство SDK for NLM и SDK Client, в состав которых, как подмножество, входят функции взаимодействия с IPX. Но использование таких инструментов разработки полностью оправдано лишь в случае написания больших и сложных сетевых прикладных программ. Если же говорить о написании программ по схеме «клиент-сервер» без использования вызовов сетевой операционной системы, то зачастую к этим программам предъявляются очень жесткие требования по использованию оперативной памяти (они должны, как правило, выполняться «резидентно»). Поэтому в данном случае становится целесообразным непосредственный вызов функций драйвера IPX через специальные прерывания и написание таких программ на языке Assembler. Именно с учетом этих требований и разрабатывались авторами различные программы, взаимодействующие с IPX. В данной статье для простоты и наглядности приводятся примеры вызова прерываний из программ на языке С.

```
void __far (*ipx_present)(void);//указатель на функцию драйвера
                                // IPX
int ipx_installed (void)
{
        union REGS reg;
        struct SREGS sreg;

        Reg.x.ax = 0x7a00;
        int86x(0x2f, &reg, &reg, &sreg);
        if(reg.h.al != 0xff) return -1;
        ipx_present = MK_FP(sreg.es, reg.x.di);
        return 1;
}
```

По такой схеме выполняется вызов функций IPX. Список функций IPX имеется в справочных руководствах по программированию в сети [3], а вызов этих функций не вызывает существенных затруднений.

Более подробно рассмотрим структуру прикладных программ. Обычно одна из рабочих станций выполняет запросы на выполнение каких-либо действий от других станций. При такой схеме взаимодействия компьютер, выполняющий запросы, является сервером, а запросившая станция - клиентом. Сервер и клиент при необходимости могут поменяться местами.

Для создания программ-серверов и программ-клиентов необходимо выполнить две задачи:

- инициализация сервера и клиента;
- прием-передача пакетов данных.

Для инициализации программ сервера и клиента помимо проверки наличия драйвера IPX и определения точки входа в его API необходимо выполнить дополнительные действия. Первое - это открыть «гнезда» (возможно не одно) для программ сервера и клиента. «Гнездо» может быть постоянным или временным. Постоянное «гнездо» должно быть закрыто по завершении программы вызовом специальной функции IPX, временные «гнезда» закрываются автоматически по завершении программы. Пример открытия временного гнезда (язык С и встроенный Ассемблер):

```
int open_socket(unsigned int socket)
{
        int num_socket;
        __asm{ mov dx, socket
               mov bx, 0000h
               mov ax, 00h }

        ipx_present();

        __asm{ mov ah, 00h
               mov num_socket, ax }
        return num_socket;
}
```

Чтобы закрыть «гнездо», можно использовать следующую функцию:

```
void close_socket(unsigned int socket)
{
        __asm{ mov bx, 0001h
               mov dx, socket }
        ipx_present();
}
```

Чтобы послать запрос от клиента к серверу, необходимо знать его сетевой адрес - адрес сети, адрес рабочей станции в сети. Для того чтобы можно было установить взаимодействие между рабочими станциями без использования файл-сервера, необходимо в качестве адреса рабочей станции указать специальное значение FFFFFFFFFFFFh. Это так называемый широковещательный запрос. Такой запрос принимают все программы на всех станциях сети, примет его и программа-сервер. Сервер, в свою очередь, может по полученному запросу послать клиенту адрес. Пакет, переданный по адресу FFFFFFFFFFFFh, будет принят, даже если файл-сервер отключен или его вовсе нет. Поэтому способ определения сетевого адреса через запрос по всей сети наиболее универсален.

Прикладные программы не могут напрямую работать с драйвером сетевого адаптера. Все запросы программа направляет драйверу IPX, который и обращается к драйверу сетевого адаптера. Для приема или передачи пакета прикладная программа должна подготовить пакет данных, сформировать его заголовок и построить так называемый блок управления событием ECB (Event Control Block). В этом блоке должны задаваться адресная информация для передачи пакета, адрес передаваемого пакета в оперативной памяти и некоторая дополнительная информация. Подготовив блок ECB, прикладная программа передает его адрес соответствующей функции драйвера IPX для выполнения операции

приема или передачи пакета. Функции IPX, принимающие или передающие пакет, не выполняют операции ожидания полного завершения приема или передачи. Они сразу возвращают управление вызвавшей их программе. После завершения операции приема/передачи в соответствующем поле блока ECB устанавливается признак. Программа может периодически опрашивать ECB для обнаружения признака завершения операции. Есть и другая возможность проверки выполнения операции приема/передачи. В блоке ECB можно указать адрес процедуры, которая будет вызвана при завершении операции передачи данных. При таком способе нет необходимости тратить время на периодическую проверку блока ECB. Такова приблизительная схема использования драйвера IPX для организации взаимодействия между рабочими станциями локальной вычислительной сети по схеме «клиент-сервер».

Статья не претендует на руководство по написанию подобных программ, а лишь указывает наиболее существенные моменты при разработке сетевых программ, не использующих системные вызовы файлового сервера. Что же касается законченных разработок, использующих протоколы обмена IPX/SPX, то авторами создан целый ряд подобных программ, которые с успехом применяются в локальных сетях различных учреждений нашего региона и Украины.

SUMMARY

The article is devoted to aspects of using data exchange IPX net protocol for organizing interaction between workstations without calling server. Practical recommendations for writing low-level functions to work with IPX is geven.

СПИСОК ЛИТЕРАТУРЫ

1. А.В.Фролов, Г.В.Фролов. Локальные сети персональных компьютеров. Использование протоколов IPX, SPX, NETBIOS. М.: Диалог-МИФИ, 1993.
2. Барри Нанс. Программирование в локальных сетях. Пермь: 1992.
3. NOVELL Netware Version 3.11. Novell Inc. USA, 1992.
4. А.Ермолов. Что такое IPX и как его использовать в своих целях //Монитор, 1992, №7,8.

Поступила в редколлегию 18 января 1995г.

УДК 681.3.06

КОМБИНИРОВАНИЕ МОДЕЛЕЙ ДАННЫХ КАК РАДИКАЛЬНЫЙ ПУТЬ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ ПРОГРАММ ОБРАБОТКИ БАЗ ДАННЫХ

Любчак В.А., доц., Лученко С.В., инж.

В этой статье на основе сравнительного анализа достоинств и недостатков реляционной и сетевой моделей баз данных (БД) указан один из возможных путей повышения эффективности программ, обрабатывающих базы данных.

Рассматривается распространенный случай представления базы данных в виде набора связанных файлов. При этом становится актуальной проблема выбора модели данных. Модель данных (или модель базы данных) определяет каким образом осуществляются взаимосвязи между записями связанных файлов внутри базы данных. От удачного выбора модели зависит эффективность работы создаваемой прикладной программы, затраты на ее разработку.

В реляционной модели база данных рассматривается как множество двумерных таблиц. Столбцы таблицы соответствуют полям данных, а строки таблицы - записям. При разделении информации на несколько таблиц объем избыточной информации уменьшается. Чтобы сохранить