МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Казанский национальный исследовательский технический университет

им. А.Н. Туполева-КАИ»

Институт компьютерных технологий и защиты информации

Кафедра Прикладной математики и Информатики имени Ю. В. Кожевникова

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
К КУРСОВОЙ РАБОТЕ**

**по дисциплине** «Компьютерные сети»

**Тема работы**: «Построение и реализация шлюза»

Выполнил:

студент группы 4410

Нигамадянов Ф.М.

Проверил:

доцент каф. ПМИ к.т.н.  
Урахчинский И. Н.

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

Казань 2023

Оглавление

[Введение 3](#_Toc122316774)

[1. Постановка задачи 4](#_Toc122316775)

[2. Описание разработанного алгоритма 5](#_Toc122316776)

[3. Подробное описание программы 8](#_Toc122316777)

[4. Описание сообщений об ошибках, выдаваемых разработанной программой 11](#_Toc122316778)

[5. Руководство пользователя 12](#_Toc122316779)

[6. Листинг программы 15](#_Toc122316780)

[6.1. Отправитель (NetBIOS-каналы) 15](#_Toc122316781)

[6.2. Шлюз 17](#_Toc122316782)

[6.3. Получатель 1 (IPX) 23](#_Toc122316783)

[6.4. Получатель 2 (NetBIOS-датаграммы) 26](#_Toc122316784)

# Введение

Сетевой шлюз (Gateway) – это аппаратный маршрутизатор или программное обеспечение для сопряжения компьютерных сетей, использующих разные протоколы (например, локальной и глобальной).

Основная задача сетевого шлюза – конвертировать протоколы одного типа физической среды в протоколы другой физической среды (сети). Таким образом, сетевой шлюз может с одной стороны принять пакет, сформатированный под один протокол и конвертировать в пакет другого протокола перед отправкой в другой сегмент сети.

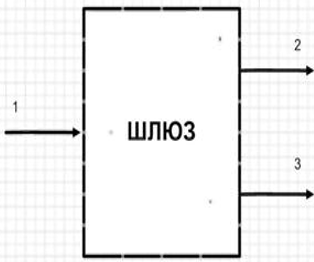
Сетевые шлюзы могут быть аппаратным решением, программным обеспечением или тем и другим вместе, но обычно это программное обеспечение, установленное на роутер или компьютер. Обычно они работают медленнее, чем сетевые мосты, коммутаторы и обычные маршрутизаторы.

В рамках данной курсовой работы был создан программный сетевой шлюз, который работает с протоколами NetBIOS-каналы, IPX и NetBIOS-датаграммы.

# Постановка задачи

Вариант 12

Необходимо спроектировать и реализовать шлюз, который принимает сообщение от отправителя по протоколу NetBIOS-каналы, формирует ответы и, используя безусловный тип передачи, отсылает их получателям по протоколам IPX и NetBIOS-датаграммы.

  
Рис. 1. Тип шлюза:  
1 –NetBIOS-каналы; 2 – IPX; 3 – NetBIOS-датаграммы. Тип передачи: безусловная

# Описание разработанного алгоритма

Для решения поставленной задачи в Oracle VM VirtualBox были созданы две виртуальные машины с операционными системами Windows XP: «vm\_net1» и «vm\_net\_2».

Далее были разработаны следующие программы в среде разработки Turbo Pascal: «NetBIOS\_Channel\_Client», «Gateway», «IPX\_Server» и «NetBIOS\_Datagram\_Server».

На виртуальной машине «vm\_net1» запускается «NetBIOS\_Channel\_Client», который отправляет сообщения в шлюз. На виртуальной машине «vm\_net2» запускается программа-шлюз «Gateway», принимающая сообщение отправителя и передающая его получателям – «NetBIOS\_Datagram\_Server» и «NetBIOS\_Channel\_Server», размещенным на первой машине.

Описание алгоритмов работы программ:

*Программа «NetBIOS\_Channel\_Client»*:

1. Удаление имени из таблицы имен рабочей станции.
2. Добавление имени в таблицу имен рабочей станции.
3. Установление канала со шлюзом с выводом номера канала на экран. При успешном исходе переход к п.4.
4. Цикл отправки сообщения:
   1. Ожидание ввода сообщения.
   2. Отправка сообщения в шлюз по протоколу NetBIOS-каналы и вывод отправляемого сообщения.
   3. Выход из цикла при вводе пользователем сообщения «exit».
5. Закрытие созданного канала с выводом соответствующего сообщения на экран.
6. Ожидание нажатия клавиши.

*Программа «Gateway»*:

1. Подготовка к приему сообщений по протоколу NetBIOS-каналы:
   1. Удаление имени из таблицы имен рабочей станции.
   2. Добавление имени в таблицу имен рабочей станции.
   3. Установление канала с отправителем с выводом номера канала на экран.
2. Подготовка для отправки сообщений по протоколу IPX:
   1. Проверка загрузки драйвера IPX:
      1. При успешном исходе: вывод соответствующего сообщения на экран и переход к п.2.2.
      2. При провальном исходе: вывод сообщения об ошибке.
   2. Открытие сокета:
      1. При успешном исходе: вывод соответствующего сообщения на экран.
      2. При провальном исходе: вывод сообщения об ошибке.
3. Подготовка для отправки сообщений по протоколу NetBIOS-датаграммы:
   1. Удаление имени из таблицы имен рабочей станции.
   2. Добавление имени в таблицу имен рабочей станции.
4. Цикл приема сообщения и отправки их получателям:
   1. Прием сообщения по протоколу NetBIOS-каналы.
   2. Вывод принятого сообщения на экран.
   3. Выход из цикла при вводе пользователем сообщения «exit».
5. Закрытие канала, созданного по протоколу NetBIOS-каналы, с выводом соответствующего сообщения на экран.
6. Закрытие сокета с выводом соответствующего сообщения на экран.
7. Ожидание нажатия клавиши.

*Программа «IPX\_Server»*:

1. Проверка загрузки драйвера IPX:
   1. При успешном исходе: вывод соответствующего сообщения на экран и переход к п.2.
   2. При провальном исходе: вывод сообщения об ошибке.
2. Открытие сокета:
   1. При успешном исходе: вывод соответствующего сообщения на экран и переход к п.3.
   2. При провальном исходе: вывод сообщения об ошибке.
3. Цикл приема сообщения:
   1. Вывод принятого по протоколу IPX от шлюза сообщения.
   2. Выход из цикла при получении сообщения «exit».
4. Закрытие сокета с выводом соответствующего сообщения на экран.
5. Ожидание нажатия клавиши.

*Программа «NetBIOS\_Datagram\_Server»*:

1. Удаление имени из таблицы имен рабочей станции.
2. Добавление имени в таблицу имен рабочей станции.
3. Цикл получения сообщения:
   1. Вывод принятого по протоколу NetBIOS-датаграммы от шлюза сообщения.
   2. Выход из цикла при получении сообщения «exit».
4. Ожидание нажатия клавиши.

# Подробное описание программы

Отправитель

В качестве отправителя сообщений выступает программа «NetBIOS\_Channel\_Client».

При запуске программы начинается настройка протокола NetBIOS: формируется NCB блок. В его поле Cmd записываются команды на выполнение, каждая команда требует настройки блока NCB. Сначала происходит удаление имени из таблицы имен рабочей станции (Cmd=31h, DeleteNameW) и добавление имени в таблицу имен рабочей станции (Cmd=30h, AddNameW). Далее осуществляется создание канала между отправителем и шлюзом при помощи команды CallW (Cmd=10h). После выполнения команды мы получаем номер канала, который будет использоваться для отправки сообщений. Отправка сообщений осуществляется командой SendW (Cmd=14h). В цикле пользователь вводит текст сообщения, размер которого не превышает 16 символов. После успешной отправки сообщение выводится на экран. Ввод слова «exit» приводит к выходу из цикла и закрытию канала с помощью команды HangUpW (Сmd=12h). Далее ожидается нажатие на клавишу, которое приведет к закрытию программы.

Шлюз

В качестве шлюза выступает программа «Gateway».

При запуске программы осуществляется подготовка к приему сообщений по протоколу NetBIOS-каналы: формируется NCB блок. Сначала происходит удаление имени из таблицы имен рабочей станции (Cmd=31h, DeleteNameW) и добавление имени в таблицу имен рабочей станции (Cmd=30h, AddNameW). Далее осуществляется создание канала между получателем и шлюзом при помощи команды ListenW (Cmd=11h). После выполнения команды мы получаем номер канала, который будет использоваться для приема сообщений. Далее необходимо подготовиться к отправке сообщений получателям, начнем с IPX.

Сначала формируется ECB блок и пакет Packet (выполнение команд осуществляется за счет драйвера и заполнения соответствующих полей ECB и Packet). Сначала необходимо проверить загрузку драйвера IPX. Для этого необходимо загрузить в регистр АХ значение 7А00h и вызвать прерывание INТ 2Fh. Если после возврата из прерывания в регистре АL будет значение FFh, драйвер IPX загружен. При наличии драйвера открывается сокет с помощью команды IPXOpenSocket (BX=00h). Далее настраиваемся на отправку сообщений по протоколу NetBIOS-датаграммы.

Всё начинается с настройки протокола NetBIOS: формируется NCB блок. Сначала происходит удаление имени из таблицы имен рабочей станции (Cmd=31h, DeleteNameW) и добавление имени в таблицу имен рабочей станции (Cmd=30h, AddNameW).

Если NetBIOS-канал был успешно создан, то следующим этапом после настройки является организация цикла для получения сообщений и отправки их получателям. Прием сообщений по протоколу NetBIOS-каналы осуществляется посредством команды ReceiveAnyW (Cmd=16h). Далее принятое сообщение отображается на экране. Первому получателю сообщение необходимо отправить по протоколу IPX командой IPXSendPacket (BX=03h), а второму получателю по протоколу NetBIOS-датаграммы командой SendDatagramW (Cmd=20h). Цикл приема сообщений обрывается при приеме сообщения «exit». После выхода осуществляется закрытия канала NetBIOS с помощью команды HangUpW (Сmd=12h). Также закрывается сокет посредством команды IPXCloseSocket (BX=01h). Далее ожидается нажатие на клавишу, которое приведет к закрытию программы.

Получатель 1

В качестве первого получателя сообщений выступает программа «IPX\_Server».

Осуществляется настройка протокола IPX: формируется ECB блок и пакет Packet (выполнение команд осуществляется за счет драйвера и заполнения соответствующих полей ECB и Packet). Сначала необходимо проверить загрузку драйвера IPX. Для этого необходимо загрузить в регистр АХ значение 7А00h и вызвать прерывание INТ 2Fh. Если после возврата из прерывания в регистре АL будет значение FFh, драйвер IPX загружен. При наличии драйвера открывается сокет с помощью команды IPXOpenSocket (BX=00h). Если сокет успешно открылся, организовывается цикл по приему сообщений по протоколу IPX. Это осуществляется с помощью команды IPXListenForPacket (BX=04h). Далее принятое сообщение отображается на экране. Прием слова «exit» приводит к выходу из цикла и закрытию сокета посредством команды IPXCloseSocket (BX=01h). Далее ожидается нажатие на клавишу, которое приведет к закрытию программы.

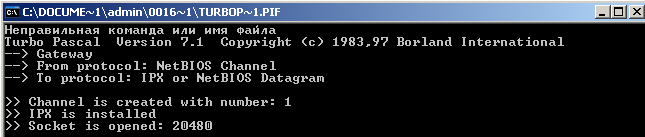
Получатель 2

В качестве второго получателя сообщений выступает программа «NetBIOS\_Datagram\_Server».

При запуске программы начинается настройка протокола NetBIOS: формируется NCB блок. Сначала происходит удаление имени из таблицы имен рабочей станции (Cmd=31h, DeleteNameW) и добавление имени в таблицу имен рабочей станции (Cmd=30h, AddNameW). Прием датаграмм (сообщений) осуществляется командой ReceiveDatagramW (Cmd=21h). В цикле осуществляется вывод принятых сообщений. Прием слова «exit» приводит к выходу из цикла. Далее ожидается нажатие на клавишу, которое приведет к закрытию программы.

# Описание сообщений об ошибках, выдаваемых разработанной программой

Вывод ошибок в разработанной программе связан с созданием NetBIOS-канала между отправителем и шлюзом, проверками наличия драйвера IPX и открытием соответствующих сокетов (рис. 2).

  
Рис. 2. Информационные сообщения в шлюзе о созданном NetBIOS-канале, наличии драйвера IPX и открытии сокета

Аналогичные информационные сообщения отображаются у отправителя и первого получателя (рис. 3-4).

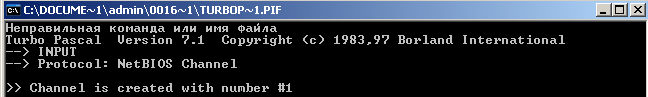


Рис. 3. Окно «NetBIOS\_Channel\_Client» (отправитель)

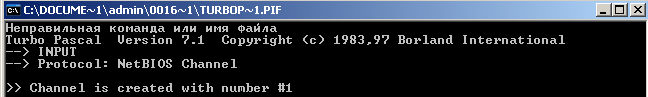


Рис. 4. Окно «IPX\_Server» (первый получатель)

При отсутствии драйвера IPX и неоткрытом сокете будут выводиться сообщения «IPX is not installed» и «Socket is not opened» соответственно. А провальная попытка создать NetBIOS-канал сопровождается сообщением «Channel is not created».

# Руководство пользователя

Запускаем виртуальные машины «vm\_net1» и «vm\_net2». На первой виртуальной машине открываем файлы NetBIOS\_Channel\_Client.PAS, IPX\_Server.PAS и NetBIOS\_Datagram\_Server.PAS, на второй – файл Gateway.PAS. Запускаем всё в следующем порядке: 1. «IPX\_Server» и «NetBIOS\_Datagram\_Server»; 2. Шлюз; 3. «NetBIOS\_Channel\_Client». В итоге получаем окна, которые представлены на рисунках 5-8.

  
Рис. 5. Окно «NetBIOS\_Channel\_Client»

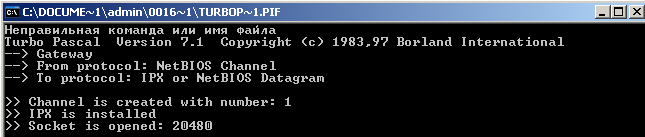
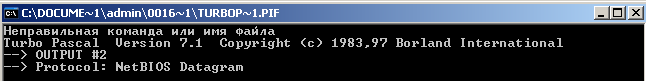
Рис. 6. Окно «Gateway»  


Рис. 7 Окно «IPX\_Server»

  
Рис. 8. Окно «NetBIOS\_Datagram\_Server»

Далее пользователь может в соответствующее окно вводить сообщения и наблюдать их перенаправление через шлюз получателям. Отправка сообщения «exit» приведет к завершению работы программы (рис. 9-12).

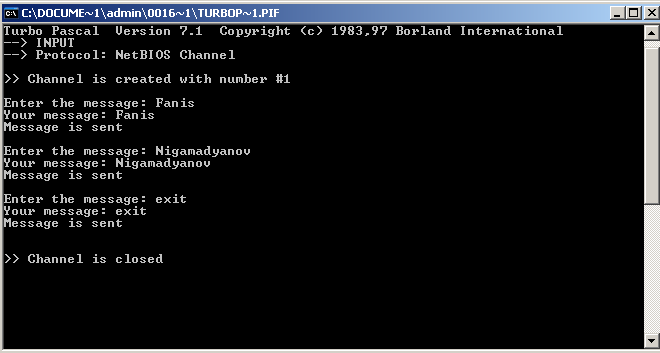


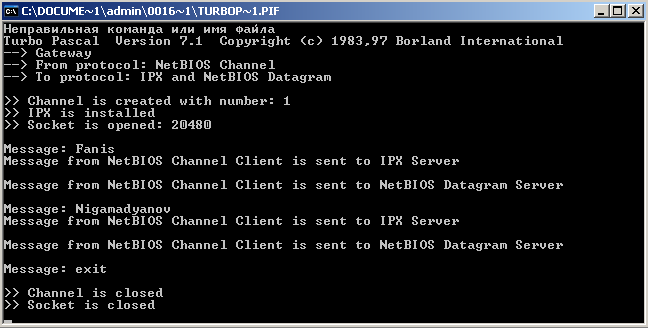
Рис. 9. Окно «NetBIOS\_Channel\_Client»  


Рис. 10. Окно «Gateway»

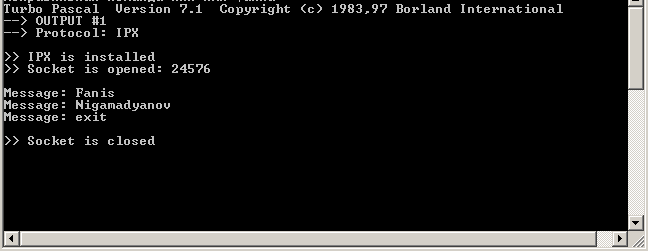
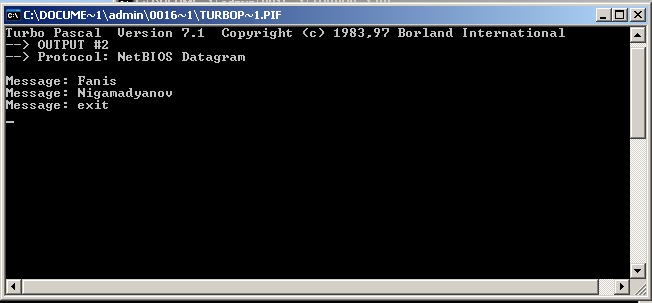


Рис. 11. Окно «IPX\_Server»

  
Рис. 12. Окно «NetBIOS\_Datagram\_Server»

# Листинг программы

# Отправитель (NetBIOS-каналы)

program NetBIOS\_Channel\_Client;

uses dos;

type

ncb=record

Cmd: byte;

CCode: byte;

LocalSessionNumber: byte;

NetworkNameNumber: byte;

BufOfs: word;

BufSeg: word;

Size: word;

CallName: array[1..16] of char;

OurName: array[1..16] of char;

ReceiveTimeout: byte;

SendTimeout: byte;

PostRoutineOfs: word;

PostRoutineSeg: word;

AdapterNumber: byte;

FinalCCode: byte;

Reserver: array[1..14] of char;

end;

var

blockNCB: ncb;

channelNumber: byte;

reg: registers;

i: integer;

buffer: string;

BEGIN

writeln('--> INPUT');

writeln('--> Protocol: NetBIOS Channel');

writeln;

{Delete name}

with blockNCB do begin

Cmd:=$31;

OurName:='netbiosclient\_in';

PostRoutineOfs:=0;

PostRoutineSeg:=0;

AdapterNumber:=3;

end;

reg.es:=seg(blockNCB);

reg.bx:=ofs(blockNCB);

intr($5C,reg);

{Add name}

with blockNCB do begin

Cmd:=$30;

OurName:='netbiosclient\_in';

PostRoutineOfs:=0;

PostRoutineSeg:=0;

AdapterNumber:=3;

end;

reg.es:=seg(blockNCB);

reg.bx:=ofs(blockNCB);

intr($5C,reg);

{Call for channel creating}

with blockNCB do begin

Cmd:=$10;

OurName:='netbiosclient\_in';

CallName:='netbiosserver\_in';

ReceiveTimeout:=10;

SendTimeout:=10;

PostRoutineOfs:=0;

PostRoutineSeg:=0;

AdapterNumber:=3;

end;

reg.es:=seg(blockNCB);

reg.bx:=ofs(blockNCB);

intr($5C,reg);

channelNumber:=blockNCB.LocalSessionNumber;

if channelNumber<>0 then

begin

writeln('>> Channel is created with number #', channelNumber);

writeln;

{Send message}

with blockNCB do begin

Cmd:=$14;

LocalSessionNumber:=channelNumber;

BufSeg:=seg(buffer);

BufOfs:=ofs(buffer);

Size:=16;

PostRoutineOfs:=0;

PostRoutineSeg:=0;

AdapterNumber:=3;

end;

reg.es:=seg(blockNCB);

reg.bx:=ofs(blockNCB);

repeat

write('Enter the message: ');

readln(buffer);

intr($5c,reg);

writeln('Your message: ', buffer);

writeln('Message is sent');

writeln;

until buffer='exit';

{Closing of channel'}

with blockNCB do begin

Cmd:=$12;

LocalSessionNumber:=channelNumber;

PostRoutineOfs:=0;

PostRoutineSeg:=0;

AdapterNumber:=3;

end;

reg.es:=seg(blockNCB);

reg.bx:=ofs(blockNCB);

intr($5C,reg);

writeln;

writeln('>> Channel is closed');

end

else

writeln('>> Channel is not created');

readln;

END.

# Шлюз

program Gateway;

uses dos;

type

ncb=record

Cmd: byte;

CCode: byte;

LocalSessionNumber: byte;

NetworkNameNumber: byte;

BufOfs: word;

BufSeg: word;

Size: word;

CallName: array[1..16] of char;

OurName: array[1..16] of char;

ReceiveTimeout: byte;

SendTimeout: byte;

PostRoutineOfs: word;

PostRoutineSeg: word;

AdapterNumber: byte;

FinalCCode: byte;

Reserver: array[1..14] of char;

end;

packet=record

CheckSum: word;

Length: word;

TransportControl: byte;

PacketType: byte;

DestNetwork: array[1..4] of byte;

DestNode: array[1..6] of byte;

DestSocket: word;

SourceNetwork: array[1..4] of byte;

SourceNode: array[1..6] of byte;

SourceSocket: word;

Data: string;

end;

ecb=record

Link: array[1..2] of word;

ESRAddrOfs: word;

ESRAddrSeg: word;

InUse: byte;

CCode: byte;

Socket: word;

IPXWork: array[1..2] of word;

DrvWork: array[1..6] of word;

ImmAddr: array[1..6] of byte;

FragCnt: word;

Addr1Ofs: word;

Addr1Seg: word;

Size1: word;

Addr2Ofs: word;

Addr2Seg: word;

Size2: word;

end;

var

blockNCB\_D, blockNCB\_Ch: ncb;

reg\_D, reg\_Ch: registers;

buffer: string;

channelNumber: byte;

nameNumber: byte;

pac: packet;

ECBblock: ecb;

regs: registers;

i: integer;

dx1: word;

al1: byte;

driver: array[1..2] of word;

\_seg,\_ofs: word;

procedure NetBIOS\_Channel\_Server;

begin

{Delete name}

with blockNCB\_Ch do begin

Cmd:=$31;

OurName:='netbiosserver\_in';

PostRoutineOfs:=0;

PostRoutineSeg:=0;

AdapterNumber:=3;

end;

reg\_Ch.es:=seg(blockNCB\_Ch);

reg\_Ch.bx:=ofs(blockNCB\_Ch);

intr($5C,reg\_Ch);

{Add name}

with blockNCB\_Ch do begin

Cmd:=$30;

OurName:='netbiosserver\_in';

PostRoutineOfs:=0;

PostRoutineSeg:=0;

AdapterNumber:=3;

end;

reg\_Ch.es:=seg(blockNCB\_Ch);

reg\_Ch.bx:=ofs(blockNCB\_Ch);

intr($5C,reg\_Ch);

{Listen for channel creating}

with blockNCB\_Ch do begin

Cmd:=$11;

OurName:='netbiosserver\_in';

CallName:='netbiosclient\_in';

ReceiveTimeout:=10;

SendTimeout:=10;

PostRoutineOfs:=0;

PostRoutineSeg:=0;

AdapterNumber:=3;

end;

reg\_Ch.es:=seg(blockNCB\_Ch);

reg\_Ch.bx:=ofs(blockNCB\_Ch);

intr($5C,reg\_Ch);

channelNumber:=blockNCB\_Ch.LocalSessionNumber;

if channelNumber<>0 then

writeln('>> Channel is created with number: ', channelNumber)

else

writeln('>> Channel is not created');

end;

procedure IPX\_Client;

begin

{Check IPX}

regs.AX:=$7A00;

intr($2F, regs);

if regs.al=$FF then begin

writeln('>> IPX is installed');

{Open Socket}

driver[1]:=regs.di;

driver[2]:=regs.es;

asm

mov bx,00h

mov al,00h

mov dx,5000h

call dword ptr driver

mov al1, al

mov dx1, dx

end;

regs.al:=al1;

if regs.al=$00 then begin

writeln('>> Socket is opened: ',dx1);

{Preparing to send packet}

with ECBblock do begin

ESRAddrOfs:=0;

ESRAddrSeg:=0;

Socket:=$5000;

FragCnt:=1;

Addr1Ofs:=ofs(pac);

Addr1Seg:=seg(pac);

Size1:=200;

end;

with pac do begin

PacketType:=4;

DestSocket:=$6000;

end;

for i:= 1 to 4 do begin

pac.DestNetwork[i]:=0;

end;

for i:=1 to 6 do begin

pac.DestNode[i]:=$FF;

end;

\_seg:=seg(ECBblock);

\_ofs:=ofs(ECBblock);

driver[1]:=regs.di;

driver[2]:=regs.es;

end

else

writeln('>> Socket is not opened');

end

else

writeln('>> IPX is not installed');

end;

procedure NetBIOS\_Datagram\_Client;

begin

{Delete name}

with blockNCB\_D do begin

Cmd:=$31;

OurName:='netbiosclientout';

PostRoutineOfs:=0;

PostRoutineSeg:=0;

AdapterNumber:=3;

end;

reg\_D.es:=seg(blockNCB\_D);

reg\_D.bx:=ofs(blockNCB\_D);

intr($5C,reg\_D);

{Add name}

with blockNCB\_D do begin

Cmd:=$30;

OurName:='netbiosclientout';

PostRoutineOfs:=0;

PostRoutineSeg:=0;

AdapterNumber:=3;

end;

reg\_D.es:=seg(blockNCB\_D);

reg\_D.bx:=ofs(blockNCB\_D);

intr($5C,reg\_D);

nameNumber:= blockNCB\_D.NetworkNameNumber;

end;

procedure SendToOutput1;

begin

{Send message}

pac.Data:=buffer;

asm

mov bx,03h

mov es,\_seg

mov si,\_ofs

call dword ptr driver

end;

if buffer <> 'exit' then begin

writeln('Message from NetBIOS Channel Client is sent to IPX Server');

writeln;

end;

end;

procedure SendToOutput2;

begin

{Send datagram}

with blockNCB\_D do begin

Cmd:=$20;

NetworkNameNumber:=nameNumber;

CallName:='netbiosserverout';

PostRoutineOfs:=0;

PostRoutineSeg:=0;

Size:=16;

BufSeg:=seg(buffer);

BufOfs:=ofs(buffer);

AdapterNumber:=3;

end;

reg\_D.es:=seg(blockNCB\_D);

reg\_D.bx:=ofs(blockNCB\_D);

intr($5C,reg\_D);

if buffer <> 'exit' then begin

writeln('Message from NetBIOS Channel Client is sent to NetBIOS Datagram Server');

writeln;

end;

end;

BEGIN

writeln('--> Gateway');

writeln('--> From protocol: NetBIOS Channel');

writeln('--> To protocol: IPX and NetBIOS Datagram');

writeln;

NetBIOS\_Channel\_Server;

IPX\_Client;

NetBIOS\_Datagram\_Client;

writeln;

while (true) do

begin

{Receive message}

if channelNumber<>0 then

begin

with blockNCB\_Ch do begin

Cmd:=$16;

LocalSessionNumber:=channelNumber;

NetworkNameNumber:=$FF;

PostRoutineOfs:=0;

PostRoutineSeg:=0;

Size:=16;

BufSeg:=seg(buffer);

BufOfs:=ofs(buffer);

AdapterNumber:=3;

end;

reg\_Ch.es:=seg(blockNCB\_Ch);

reg\_Ch.bx:=ofs(blockNCB\_Ch);

intr($5C,reg\_Ch);

writeln('Message: ', buffer);

if regs.al=$00 then

SendToOutput1;

SendToOutput2;

if buffer = 'exit' then

break;

end;

end;

writeln;

if channelNumber<>0 then

begin

{Close channel}

with blockNCB\_Ch do begin

Cmd:=$12;

LocalSessionNumber:=channelNumber;

PostRoutineOfs:=0;

PostRoutineSeg:=0;

AdapterNumber:=3;

end;

reg\_Ch.es:=seg(blockNCB\_Ch);

reg\_Ch.bx:=ofs(blockNCB\_Ch);

intr($5C,reg\_Ch);

writeln('>> Channel is closed');

end;

if regs.al=$00 then

begin

{Close socket}

driver[1]:=regs.di;

driver[2]:=regs.es;

asm

mov bx,01h

mov al,00h

mov dx,5000h

call dword ptr driver

end;

writeln('>> Socket is closed');

end;

readln;

END.

# Получатель 1 (IPX)

program IPX\_Server;

uses dos;

type

packet=record

CheckSum: word;

Length: word;

TransportControl: byte;

PacketType: byte;

DestNetwork: array[1..4] of byte;

DestNode: array[1..6] of byte;

DestSocket: word;

SourceNetwork: array[1..4] of byte;

SourceNode: array[1..6] of byte;

SourceSocket: word;

Data: string;

end;

ecb=record

Link: array[1..2] of word;

ESRAddrOfs: word;

ESRAddrSeg: word;

InUse: byte;

CCode: byte;

Socket: word;

IPXWork: array[1..2] of word;

DrvWork: array[1..6] of word;

ImmAddr: array[1..6] of byte;

FragCnt: word;

Addr1Ofs: word;

Addr1Seg: word;

Size1: word;

Addr2Ofs: word;

Addr2Seg: word;

Size2: word;

end;

var

pac: packet;

ECBblock: ecb;

regis: registers;

i: integer;

driver: array[1..2] of word;

dx1: word;

al1: byte;

\_seg,\_ofs: word;

message: string;

BEGIN

writeln('--> OUTPUT #1');

writeln('--> Protocol: IPX');

writeln;

{Check IPX}

regis.AX:=$7A00;

intr($2F, regis);

if regis.AL=$FF then begin

writeln('>> IPX is installed');

{Open socket}

driver[1]:=regis.di;

driver[2]:=regis.es;

asm

mov bx,00h

mov al,00h

mov dx,6000h

call dword ptr driver

mov al1, al

mov dx1, dx

end;

regis.al:=al1;

if regis.al=$00 then begin

writeln('>> Socket is opened: ',dx1);

writeln;

pac.PacketType:=4;

for i:= 1 to 4 do begin

pac.DestNetwork[i]:=0;

end;

for i:=1 to 6 do begin

pac.DestNode[i]:=$FF;

end;

pac.DestSocket:=$5000;

with ECBblock do begin

ESRAddrOfs:=0;

ESRAddrSeg:=0;

Socket:=$6000;

FragCnt:=1;

Addr1Ofs:=ofs(pac);

Addr1Seg:=seg(pac);

Size1:=200;

end;

\_seg:=seg(ECBblock);

\_ofs:=ofs(ECBblock);

driver[1]:=regis.di;

driver[2]:=regis.es;

repeat

asm

mov bx,04h

mov es,\_seg

mov si,\_ofs

call dword ptr driver

end;

while true do

begin

if ECBblock.InUse=0 then

break;

end;

writeln('Message: ', pac.Data)

until pac.Data = 'exit';

writeln;

{Close socket}

driver[1]:=regis.di;

driver[2]:=regis.es;

asm

mov bx,01h

mov al,00h

mov dx,6000h

call dword ptr driver

end;

writeln('>> Socket is closed');

readln;

end

else

writeln('>> Socket is not opened');

end

else

writeln('>> IPX is not installed');

END.

# Получатель 2 (NetBIOS-датаграммы)

program NetBIOS\_Datagram\_Server;

uses dos;

type

ncb=record

Cmd: byte;

CCode: byte;

LocalSessionNumber: byte;

NetworkNameNumber: byte;

BufOfs: word;

BufSeg: word;

Size: word;

CallName: array[1..16] of char;

OurName: array[1..16] of char;

ReceiveTimeout: byte;

SendTimeout: byte;

PostRoutineOfs: word;

PostRoutineSeg: word;

AdapterNumber: byte;

FinalCCode: byte;

Reserver: array[1..14] of char;

end;

var

NCBlock: ncb;

reg: registers;

buffer: string;

BEGIN

writeln('--> OUTPUT #2');

writeln('--> Protocol: NetBIOS Datagram');

writeln;

{Delete name}

with NCBlock do begin

Cmd:=$31;

OurName:='netbiosserverout';

PostRoutineOfs:=0;

PostRoutineSeg:=0;

AdapterNumber:=3;

end;

reg.es:=seg(NCBlock);

reg.bx:=ofs(NCBlock);

intr($5C,reg);

{Add name}

with NCBlock do begin

Cmd:=$30;

OurName:='netbiosserverout';

PostRoutineOfs:=0;

PostRoutineSeg:=0;

AdapterNumber:=3;

end;

reg.es:=seg(NCBlock);

reg.bx:=ofs(NCBlock);

intr($5C,reg);

{Receive datagram}

with NCBlock do begin

Cmd:=$21;

NetworkNameNumber:=$FF;

PostRoutineOfs:=0;

PostRoutineSeg:=3;

Size:=16;

BufSeg:=seg(buffer);

BufOfs:=ofs(buffer);

AdapterNumber:=3;

end;

reg.es:=seg(NCBlock);

reg.bx:=ofs(NCBlock);

repeat

intr($5C,reg);

writeln('Message: ', buffer);

until buffer='exit';

readln;

END.