

Сеть MINIX3: протоколы и инструменты

Цилюрник О.И.

< olej@front.ru >

Редакция 3.10

от 12.02.2010

Оглавление

Аннотация.....	1
Версии системы.....	2
Структура документа.....	2
Протоколы MINIX3.....	2
SSH	4
X	6
TELNET	8
Клиент.....	8
Сервер.....	9
RLOGIN.....	10
Клиент.....	10
Сервер.....	11
RSH.....	11
FTP.....	12
Клиент.....	12
Сервер.....	13
HTTP.....	16
Клиент.....	16
Сервер.....	16
Протоколы точка-точка	19
Особенности последовательного канала в MINIX3.....	19
SLIP.....	19
CSLIP.....	22
PPP.....	22
PLIP.....	22
Инструменты сети.....	22
Конфигурирование сети MINIX3.....	22
Утилиты сети.....	23
ping.....	23
ifconfig.....	24
tcpstat.....	24
add_route.....	24
pr_routes.....	25
Как выйти во внешнюю сеть?.....	25
Разрешение сетевых имён.....	26
Приложение: Кабель для SLIP соединения компьютеров.....	27
Дополнительные источники информации.....	27

Аннотация

В этом обзоре я постараюсь перечислить существующие (и доступные) реализации сетевых протоколов под MINIX3. Относительно каждого реализованного протокола я постараюсь осветить: а). обзор реализующих

протокол программ, б). установку, в). настройку г). использование. Поскольку мы рассматриваем сетевые средства, то, чаще всего, нам нужно будет отдельно рассматривать вопрос о поддержке протокола со стороны клиента и поддержке протокола со стороны сервера. В рассмотрение будут включаться не только реализации, входящие в официальный дистрибутив, но весь спектр существующих реализаций в природе.

Версии системы

Версии MINIX3 в очень большой мере «волатильны» - разработчики часто вносят существенные изменения, даже не считая должным отражать их, зачастую, в man-ах. Вся основная часть описания отрабатывалась на стабильной версии 3.1.5 (релиз 5612), в некоторых случаях, оговоренных особо, рассмотрение ведётся на стабильной версии 3.1.6 (релиз 6084).

Структура документа

Все показанные в тексте протоколы выполнения команд сохранены прямым копированием с экрана терминала, так же как и графические скриншоты; все действия, описываемые в тексте, могут быть повторно воспроизведены.

В самом тексте, все примеры команд (скопированные с терминала) будут показываться **МОНОШИРИННЫМ** шрифтом. Кроме того, в большинстве случаев пользовательский ввод в записи команды будет показан **ЖИРНЫМ** шрифтом, а ответный вывод от системы — **ОБЫЧНЫМ**. Короткие цитаты из различных источников информации будут показываться *курсивом*.

Протоколы MINIX3

Конечно, не все сетевые протоколы, доступные из MINIX3, будут детально описываться ниже. Часть протоколов (и TCP портов) используется в качестве диагностических и тестовых, для их использования не предназначены какие-то специальные программные средства (часто они используются из клиента telnet), примерами таких есть ECHO (порт 7) и DAYTIME (порт 13), они будут использованы и показаны далее в примерах относительно telnet. Весь перечень протоколов, определённых по их символическим именам, и известных системе MINIX3, содержится в файле /etc/services, и, несмотря на его значительный объём, его стоит привести здесь полностью:

```
# cat /etc/services
#
# Network services, Internet style
#
#      @(#)services      8.1 (Berkeley) 6/9/93
#
tcpmux      1/tcp          # TCP port multiplexer (RFC1078)
echo        7/tcp
echo        7/udp
discard     9/tcp          sink null
discard     9/udp          sink null
sysstat     11/tcp         users
daytime     13/tcp
daytime     13/udp
netstat     15/tcp
qotd        17/tcp          quote
chargen     19/tcp         ttytst source
chargen     19/udp         ttytst source
```

```

ftp          21/tcp
ssh         22/tcp          #Secure Shell Login
ssh         22/udp          #Secure Shell Login
telnet      23/tcp
smtp        25/tcp          mail
time        37/tcp          timserver
time        37/udp          timserver
rtp         39/udp          resource      # resource location
nameserver  42/tcp          name          # IEN 116
whois       43/tcp          nicname
domain      53/tcp          nameserver    # name-domain server
domain      53/udp          nameserver
mtp         57/tcp          # deprecated
bootps      67/udp          # Bootstrap Protocol Server
bootpc      68/udp          # Bootstrap Protocol Client
tftp        69/udp
rje         77/tcp          netrjs
finger      79/tcp
http        80/tcp          # World Wide Web
link        87/tcp          ttylink
supdup      95/tcp
hostnames   101/tcp         hostname      # usually from sri-nic
tsap        102/tcp         # part of ISODE.
pop         110/tcp         postoffice
sunrpc      111/tcp
sunrpc      111/udp
auth        113/tcp         authentication
sftp        115/tcp
uucp-path   117/tcp
nntp        119/tcp         readnews untp # USENET News Transfer Protocol
ntp         123/udp
netbios-ns  137/tcp         # NETBIOS Name Service
netbios-ns  137/udp         # NETBIOS Name Service
netbios-dgm 138/tcp         # NETBIOS Datagram Service
netbios-dgm 138/udp         # NETBIOS Datagram Service
netbios-ssn 139/tcp         # NETBIOS Session Service
netbios-ssn 139/udp         # NETBIOS Session Service
imap        143/tcp
snmp        161/udp
snmp-trap   162/udp
#
# UNIX specific services
#
exec        512/tcp
biff        512/udp         comsat

```

```

login          513/tcp
who            513/udp      whod
shell         514/tcp      cmd          # no passwords used
syslog        514/udp
printer       515/tcp      spooler      # line printer spooler
talk          517/udp
ntalk         518/udp
route         520/udp      router routed
timed         525/udp      timeserver
tempo         526/tcp      newdate
courier       530/tcp      rpc
conference    531/tcp      chat
netnews       532/tcp      readnews
netwall       533/udp      # -for emergency broadcasts
uucp          540/tcp      uucpd        # uucp daemon
rdist         541/tcp      rdistd       # rdist daemon
kshell        544/tcp      krcmd        # Kerberos remote shell
remotefs      556/tcp      rfs_server rfs # Brunhoff remote filesystem
ingreslock    1524/tcp
#
# Kerberos (Project Athena/MIT) services
#
kerberos      750/udp      kdc          # Kerberos (server) udp
kerberos      750/tcp      kdc          # Kerberos (server) tcp
krbupdate     760/tcp      kreg        # Kerberos registration
kpasswd       761/tcp      kpwd        # Kerberos "passwd"
klogin        543/tcp      # Kerberos rlogin
rsync         873/tcp
rsync         873/udp
eklogin       2105/tcp      # Kerberos encrypted rlogin
svn           3690/tcp      # Subversion
svn           3690/udp      # Subversion
postgresql    5432/tcp      # PostgreSQL Database
postgresql    5432/udp      # PostgreSQL Database

```

Некоторые, самые используемые в MINIX3, протоколы пользовательского уровня подробно рассматриваются далее.

SSH

И клиент (ssh), и сервер (демон – sshd) устанавливаются, по требованию (по умолчанию не установлены), из репозитория LiveCD командой:

```
# packman
```

Инсталлятор packman выбирает пакеты не по именам, а по последовательным номерам, под которыми он же перечисляет пакеты пользователю (на экран). Для установки SSH необходимо установить пакеты #71(ssh) и #72(ssl). При **следующей** загрузке сервер sshd будет стартовать автоматически.

Для того, чтобы снаружи (из LAN) обратиться к серверу sshd, выполняем:

```
$ ssh -l root 192.168.3.4
root@192.168.3.4's password:
Last login: Mon Nov 23 12:17:29 2009 from 192.168.3.6
...
#
```

- приглашение # в последней показанной строке – это уже консоль MINIX3.

Примечание: во всех примерах я буду использовать сетевые адреса: 192.168.3.4 – это адрес MINIX3 со стороны Linux, 192.168.3.6 – это адрес Linux со стороны MINIX3.

Исключительно благодаря показанному выше способу подключения к консоли MINIX3 (использование SSH со стороны Linux) я имею возможность показывать здесь краткие листинги выполнения команд, а не объёмные графические скриншоты.

Обращение со стороны консоли MINIX3 посредством SSH-клиента (хотя такое бывает нужно гораздо реже):

```
# ssh -l olej 192.168.3.6
The authenticity of host '192.168.3.6 (192.168.3.6)' can't be established.
RSA key fingerprint is 8f:98:44:55:b1:6e:0b:e5:27:a7:87:f9:8a:3e:d5:43.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.3.6' (RSA) to the list of known hosts.
olej@192.168.3.6's password:
Last login: Tue Nov 24 10:15:32 2009 from 192.168.2.118
$ uname -a
Linux opos9.altron.lan 2.6.18-53.1.19.e15 #1 SMP Wed May 7 08:20:19 EDT 2008 i686 i686
i386 GNU/Linux
```

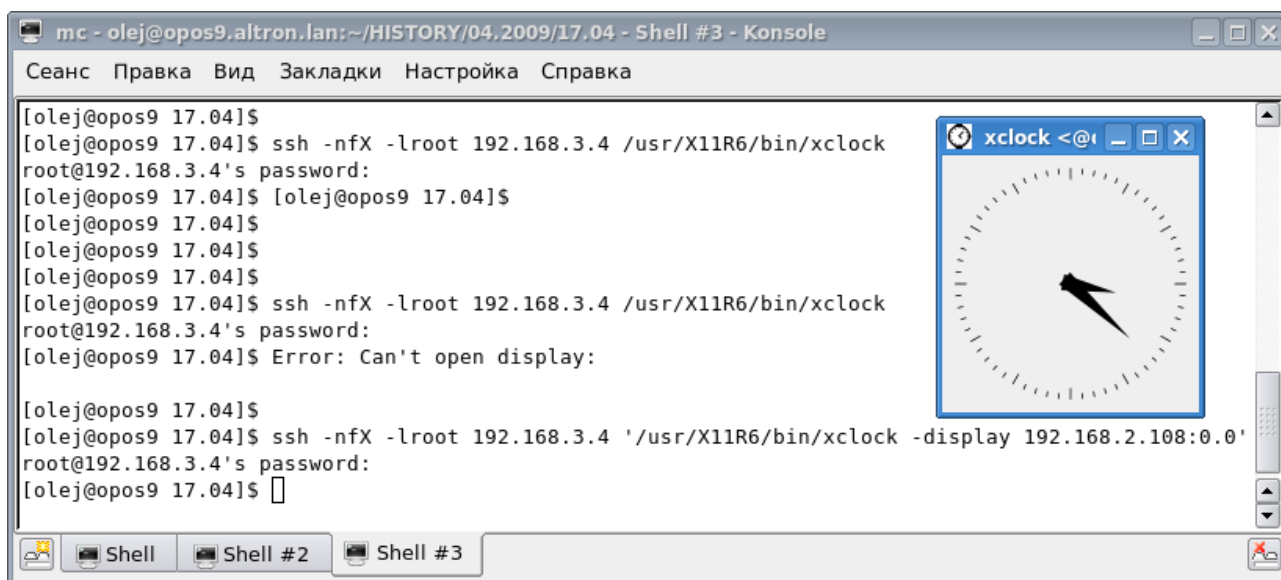
- видно, как при первом подключении к новому хосту запрашивается подтверждение на генерацию RSA-ключа (вот для этого нужен пакет ssl).

Из особенностей программ: SSH не позволит подключиться от имени пользователя, имеющего «пустой» пароль – это уже достаточное основание для того, чтобы сразу же после завершения установки MINIX3 установить пароль для root (passwd), и добавить новых пользователей (adduser).

Реализация SSH-протокола для MINIX3 «полновесна», и включает поддержку транзита сквозь себя шифрованного потока X-протокола (ключ -X), рис.1. На этом рисунке показано удалённое выполнение GUI программы (любой!) командой вила:

```
$ ssh -nfX -l<user> <host> <command>
```

Рис.1



X

Для MINIX3 есть реализация Xorg: устанавливается с дистрибутивного LiveCD, пакет X11R6 #113.

Устанавливается выполнением:

```
# packman
```

...

Стартовать (один из нескольких возможных вариантов запуска) X-сервер и сконфигурированный (изначально это будет twm) менеджер окон:

```
# xinit
```

Примечание: при первом старте X редко стартует в удовлетворительном виде – слишком высокое разрешение и другие неприятности. Настройка X – это предмет отдельного обсуждения и высокое искусство, но ниже я в 2 слова перечислю простейшие действия, которые позволят привести X к такому виду, с которого с ним можно начинать работать.

- диагностика:

```
# cd /root
```

```
# Xorg -configure
```

при этом Xorg динамически тестирует все устройства и создаёт файл `/root/xorg.conf.new` в котором записывает диагностируемую конфигурацию, диагностика весьма точная, у меня, например, относительно видеокарты:

```
Matrox Graphics
```

```
MGA G200 AGP
```

```
PCI :1:0:0
```

- проверка конфигурации, старт X-сервера без оконных менеджеров (рис.2):

```
# X -config /root/xorg.conf.new
```

- перемещение `xorg.conf.new`, Xorg в этой сборке ожидает конфигурационный файл в 2-х местах - `/etc/X11/xorg.conf` или `/usr/X11R6/etc/X11/xorg.conf` – копируем туда конфигурацию:

```
# cp /root/xorg.conf.new /etc/X11/xorg.conf
```

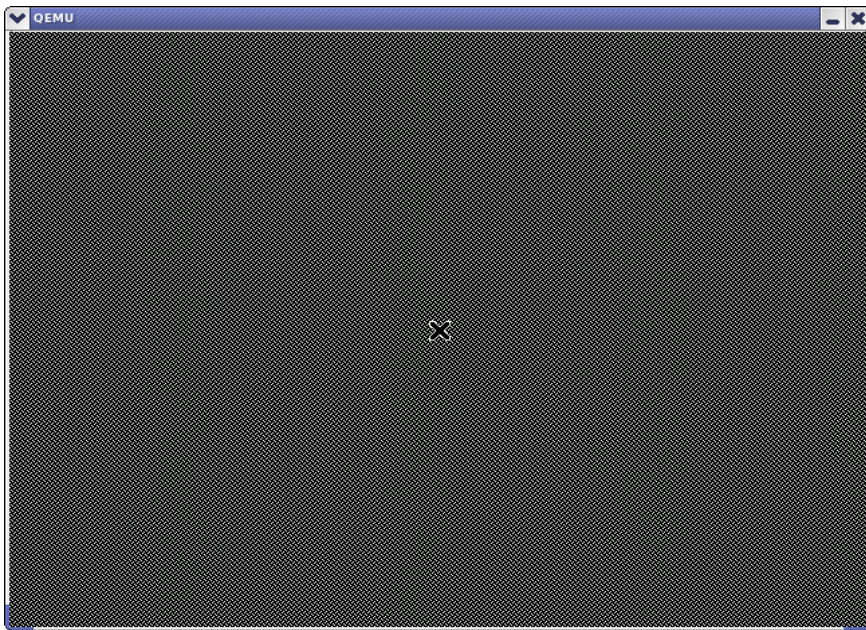


Рис.2

- есть ещё ручной способ построения `xorg.conf`, когда в диалоге мы отвечаем на множество детальных вопросов:

```
# xorgconfig
```

Способ трудоёмкий, но позволяет очень точно настроить X сервер. Комбинируя 2 способа и сливая файлы конфигурации, можно получить оптимальную настройку X.

Клиенты протокола X – это множество GUI приложений, устанавливаемых при установке Xorg, а также все GUI приложения, которые вы будете доставлять в будущем. В MINIX3 полноценная (не усечённая) реализация протокола X, и клиенты X могут выполняться без сервера X, взаимодействуя с пользователем на удалённом хосте LAN (с X сервером этого хоста). Для того, чтобы удалённый доступ к X серверу со стороны приложения был возможным, необходимо обеспечить ряд условий со стороны X сервера:

1. Разрешить на Linux X-сервере доступ от удалённого хоста (или вообще от всех хостов), далее показана проверка того, что это состоялось:

```
$ xhost +192.168.3.4
```

```
192.168.3.4 being added to access control list
```

```
$ xhost
```

```
access control enabled, only authorized clients can connect
```

```
INET:192.168.3.4
```

```
SI:localuser:olej
```

2. Проверить, что X-сервер запущен с разрешённым доступом TCP, например так:

```
$ ps ahx | grep Xorg
```

```
329 pts/10 S+ 0:00 grep Xorg
```

```
23476 tty7 Ss+ 84:49 /usr/bin/Xorg :0 -br -audit 0 -auth /var/gdm/:0.Xauth vt7
```

Если TCP доступ запрещён (а так часто и бывает после инсталляции Linux), то последняя строка (запуска Xorg) будет выглядеть подобно:

```
... /usr/bin/Xorg :0 -br -audit 0 -auth /var/gdm/:0.Xauth -nolisten tcp vt7
```

Тогда это нужно изменить. Например, можно воспользоваться менеджером:

```
# gdmsetup
```

- установить разрешение TCP доступа, и перезапустить X систему:

```
# gdm-restart
```

(не стоит здесь пугаться, что рабочая сессия здесь закроется, и будет запущена новая, начиная с начального login).

3. Войти в удалённую систему ... тем же SSH, к примеру:

```
$ ssh -l root 192.168.3.4
```

```
root@192.168.3.4's password:
```

```
Last login: Wed Nov 18 15:17:15 2009
```

4. Запустить требуемое GUI приложение в этой удалённой системе (в точности то же самое можно сделать и с консоли MINIX3 в QEMU), рис.3 :

```
# xclock -display 192.168.2.108:0.0
```

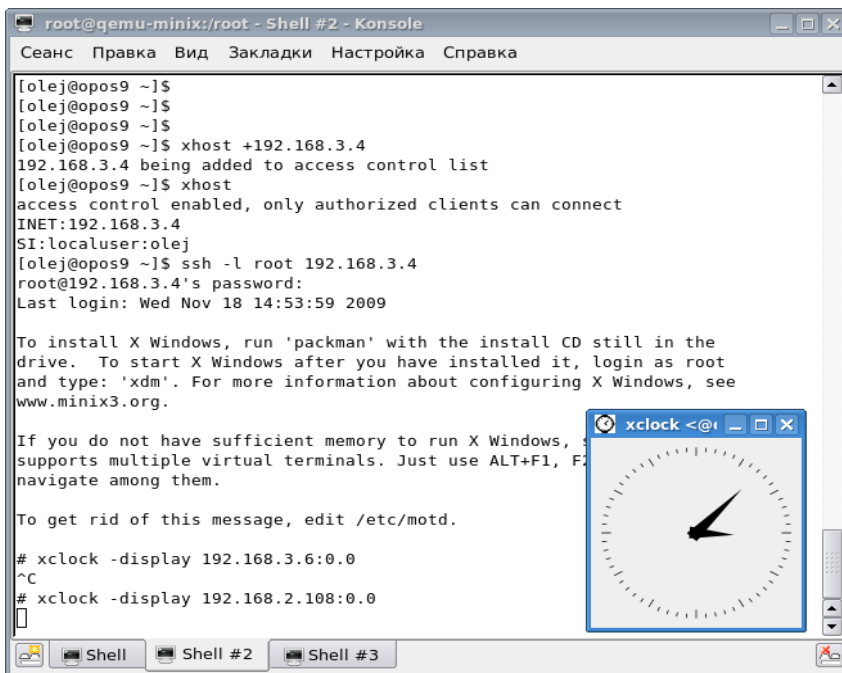


Рис. 3

TELNET

Это простейший протокол (некоторые говорят: устаревший), позволяющий осуществлять не защищённое TCP подключение к выбранному порту. При подключении к MINIX3 telnet интересен тем, что он намного быстрее SSH, так как не предусматривает шифрование на 2-х концах соединения. Кроме того, клиент telnet традиционно используется как «универсальный сетевой тестер», для подключения к различным портам и службам, как это показано в примерах ниже.

Клиент

Клиент telnet особо интересен своей возможностью подключаться к любому TCP порту, тем самым он выступает в качестве универсального сетевого тестера. В свежей установленной системе:

```
# which telnet
```

```
/usr/bin/telnet
```

- клиент telnet установлен по умолчанию, до выбора нами пакетов для установки (packman). Проверим его работоспособность (подключение со стороны MINIX3 к Linux):

```
# telnet 192.168.3.6 7
```

```
Connecting to 192.168.3.6:7...
```


Connected

1

1

2

2

3

3

^C

- это подключение к эхо-серверу (порт 7);

```
# telnet 192.168.3.6 13
```

```
Connecting to 192.168.3.6:13...
```

```
Connected
```

```
25 NOV 2009 12:38:00 EET
```

- а это подключение к серверу даты (порт 13);

Сервер

Сервер TELNET тоже присутствует в системе сразу после установки (до выполнения раскнан):

```
# cd /usr/src/commands
```

```
# ls
```

```
... telnet telnetd ...
```

Но он требует сборки (файлы присутствуют в исходном коде, но нет бинарной программы)

```
# cd /usr/src/commands/telnetd
```

```
# ls
```

```
Makefile build main.c pty.c telnet.c telnet.h telnetd.h term.c wtmp.c
```

```
# make
```

```
exec cc -c -O -D_MINIX -D_POSIX_SOURCE main.c
```

```
exec cc -c -O -D_MINIX -D_POSIX_SOURCE telnet.c
```

```
exec cc -c -O -D_MINIX -D_POSIX_SOURCE term.c
```

```
exec cc -c -O -D_MINIX -D_POSIX_SOURCE pty.c
```

```
exec cc -c -O -D_MINIX -D_POSIX_SOURCE wtmp.c
```

```
exec cc -i -o in.telnetd main.o telnet.o term.o pty.o wtmp.o
```

```
install -S 8kw in.telnetd
```

```
# make install
```

```
install -cs -o bin in.telnetd /usr/bin/in.telnetd
```

```
# ls -l /usr/bin/in.t*
```

```
-rwxr-xr-x 1 bin operator 40652 Nov 25 12:53 in.telnetd
```

Примечание: показанная здесь сборка telnetd, как и следующая далее сборка ftpd, абсолютно необходимы в MINIX3 версии 3.1.5; в версии 3.1.6 они уже собраны, и установлены в каталог:

```
# ls /usr/bin/in.*
```

```
/usr/bin/in.fingerd /usr/bin/in.ftpd /usr/bin/in.rlogind
```

```
/usr/bin/in.rshd /usr/bin/in.telnetd
```

Таким образом, прежде чем приступать к сборке серверов, проверьте, нет ли их уже установленных в вашей поставке MINIX3.

Запуск сервера `in.telnetd` в MINIX3 производится (как и всех серверных служб вида `in.*`) через демон `tcpd`:

```
# tcpd telnet in.telnetd
```

Примечание: демон `tcpd` в MINIX3 — это некоторая упрощённая аналогия суперсервера `inetd` / `xinetd`, является специфической принадлежностью MINIX3, о котором сказано:

```
# man tcpd
```

```
...
```

NOTES That daemons cannot daemonize themselves is a way in which Minix differs from most other Unix-like systems.

Обращаемся к только что запущенному серверу из Linux хоста:

```
$ telnet
```

```
telnet> open 192.168.3.4
```

```
Trying 192.168.3.4...
```

```
Connected to 192.168.3.4 (192.168.3.4).
```

```
Escape character is '^]'.
```

```
Minix Release 3 Version 1.5 (ttyp2)
```

```
qemu-minix login: root
```

```
Password:
```

```
# pwd
```

```
/root
```

```
# ls
```

```
.ashrc  .ellepro.bl  .ellepro.e  .exerc  .fonts.cache-1  .profile  .ssh
```

```
...
```

```
# exit
```

```
Connection closed by foreign host.
```

```
$
```

Примечание: вообще, все сетевые сервера после установки, здесь и далее, ищем в каталоге:

```
# ls -l /usr/bin/in.*
```

```
-rwxr-xr-x 1 bin operator 23608 Nov  5 10:01 in.fingerd
```

```
-rwxr-xr-x 1 bin operator 72876 Nov 25 11:03 in.ftpd
```

```
-rwxr-xr-x 1 bin operator 48272 Nov  5 10:01 in.rlogind
```

```
-rwxr-xr-x 1 bin operator 41756 Nov  5 10:01 in.rshd
```

```
-rwxr-xr-x 1 bin operator 40652 Nov 25 12:53 in.telnetd
```

RLOGIN

Протокол удалённого доступа (регистрации) RLOGIN — не менее ценное приобретение для организации удалённой работы с хостом в LAN, чем TELNET.

Клиент

Клиент RLOGIN присутствует, но, похоже, неработоспособен:

```
# rlogin 192.168.3.6 -l olej
```

```
unable to ioctl(NWIOTCPCONN): Connection refused
```

Сервер

С сервером RLOGIN, судя по всему, дела обстоят лучше:

```
# tcpd login in.rlogind
```

(in.rlogind сборки не требует)

Обращение к нему со стороны Linux:

```
$ /usr/bin/rlogin 192.168.3.4 -l olej
```

```
Password:
```

```
$ pwd
```

```
/home/olej
```

```
$ ls
```

```
$ uname -a
```

```
Minix qemu-minix 3 1.5 i686
```

```
$ exit
```

```
rlogin: connection closed.
```

Примечание: я не случайно указал полное имя при запуске программы rlogin — в Linux давно уже не поощряется выполнение незащищённых команд группы r*, если указать просто rlogin, то будет по умолчанию вызываться Kerberos реализация, и только после её неудачи программа прямого подключения, которую мы имели в виду. Вот протокол начала сеанса в таком случае:

```
$ rlogin 192.168.3.4 -l root
```

```
connect to address 192.168.3.4 port 543: Connection refused
```

```
Trying krb4 rlogin...
```

```
connect to address 192.168.3.4 port 543: Connection refused
```

```
trying normal rlogin (/usr/bin/rlogin)
```

```
Password:
```

RSH

Программные средства поддержки протокола RSH (Remote Shell), как клиента, так и сервера, присутствуют в MINIX3 наравне с FTP, TELNET и RLOGIN:

```
# which rsh
```

```
/usr/bin/rsh
```

```
# ls -l /usr/bin/in.*
```

```
-rwxr-xr-x 1 bin operator 23608 Nov 5 10:01 in.fingerd
```

```
-rwxr-xr-x 1 bin operator 72876 Nov 25 11:03 in.ftpd
```

```
-rwxr-xr-x 1 bin operator 48272 Nov 5 10:01 in.rlogind
```

```
-rwxr-xr-x 1 bin operator 41756 Nov 5 10:01 in.rshd
```

```
-rwxr-xr-x 1 bin operator 40652 Nov 25 12:53 in.telnetd
```

Настройку их (главным образом связанную с аутентификацией и безопасностью) я не производил, поэтому примеры использования не приводятся. Эту работу несложно, при необходимости, выполнить самостоятельно.

FTP

Клиент

Один из клиентов для FTP установлен в системе изначально (сразу после установки системы):

```
# which ftp
/usr/bin/ftp

Обращаемся из него к хосту Linux:
# ftp 192.168.3.6
220 (vsFTPd 2.0.5)
Username: olej
331 Please specify the password.
Password:
230 Login successful.
ftp>pwd
257 "/home/olej"
ftp>exit
221 Goodbye.
FTP done.
```

Ещё один клиент (ncftp) предлагается для установки (packman) с LiveCD, после установки:

```
# which ncftp
/usr/local/bin/ncftp

Снова подключаемся FTP клиентом к хосту Linux:
# ncftp
NcFTP 3.1.9 (Mar 24, 2005) by Mike Gleason (http://www.NcFTP.com/contact/).
Copyright (c) 1992-2005 by Mike Gleason.
All rights reserved.
ncftp> help
Commands may be abbreviated. 'help showall' shows hidden and unsupported
commands. 'help <command>' gives a brief description of <command>.
ascii      cat        help       lpage     open      quit      show
bgget      cd         jobs       lpwd      page      quote     site
bgput      chmod     lcd        lrename   passive   rename    type
bgstart    close     lchmod     lrm       pdir      rhelp     umask
binary     debug     lls        lrmdir    pls       rm        version
bookmark   dir       lmkdir     ls        put       rmdir
bookmarks  get       lookup     mkdir     pwd       set

For details, please see the manual ("man ncftp" at your regular shell prompt
or online at http://www.NcFTP.com/ncftp/doc/ncftp.html).
ncftp> open 192.168.3.6
Connecting to 192.168.3.6...
(vsFTPd 2.0.5)
Logging in...
Login successful.
```

```
Logged in to 192.168.3.6.
ncftp / > pwd
ftp://192.168.3.6
ncftp / > ls
pub/
ncftp / > quit
You have not saved a bookmark for this site.
Would you like to save a bookmark to:
ftp://192.168.3.6
Save? (yes/no) yes
Enter a name for this bookmark: linux
Bookmark "linux" saved.
```

Сервер

Сервер FTP присутствует в той же мере, и в том же качестве, что и сервер TELNET — он присутствует в исходных кодах, но требует сборки:

```
# cd /usr/src/commands
# ls
... ftpd200 ...
# cd ftpd200
# cat README
ftpd200 --- FTP server program for Minix 2.0
...
Ftpd is the File Transfer Protocol (FTP) server.
...
Read the Makefile to see how
the program is compiled and installed:
make (or make ftpd) -- compiles the binary
make install          -- installs /usr/bin/in.ftpd, and ftpdsh.
                      Also installs setup.anonftp script.
make installman      -- installs new ftpd.8 man page in /usr/local/man/man8
The shell script setup.anonftp sets up and verifies configuration for anonymous ftp.
# make
exec cc -c -O -D_MINIX -D_POSIX_SOURCE -m ftpd.c
exec cc -c -O -D_MINIX -D_POSIX_SOURCE -m access.c
exec cc -c -O -D_MINIX -D_POSIX_SOURCE -m file.c
exec cc -c -O -D_MINIX -D_POSIX_SOURCE -m net.c
exec cc -i -o in.ftpd ftpd.o access.o file.o net.o
install -S 8kw in.ftpd
# make install
install -cs -o bin in.ftpd /usr/bin/in.ftpd
# make installman
cp ftpd.8 /usr/man/man8
echo "You may need to run makewhatis to update man page index"
You may need to run makewhatis to update man page index
```

```
# makewhatis /usr/man/man8
# man ftpd
...
1) The user name must be in the password data base, /etc/passwd, and
not have a null password. In this case a password must be provided
by the client before any file operations may be performed.
...
```

Запуск сервера очень похож на запуск сервера TELNET, показанный раньше:

```
# tcpd ftp in.ftpd
```

Обращаемся к запущенному серверу со стороны Linux хоста:

```
$ ftp
ftp> open 192.168.3.4
Connected to 192.168.3.4.
220 FTP service (Ftpd 2.00) ready on qemu-minix at Wed, 25 Nov 2009 14:10:14 GMT
500 Command "AUTH" not recognized.
500 Command "AUTH" not recognized.
KERBEROS_V4 rejected as an authentication type
Name (192.168.3.4:olej): root
331 Password required for root.
Password:
230 User root logged in, directory /root.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (192,168,3,4,128,5).
125 File LIST okay. Opening data connection.
total 539
drwx----- 3 root operator 640 Nov 25 11:47 .
drwxr-xr-x 14 root operator 1152 Nov 18 12:05 ..
-rw-r--r-- 1 root operator 592 Nov 5 10:00 .ashrc
-rw-r--r-- 1 root operator 300 Nov 5 10:00 .ellepro.b1
-rw-r--r-- 1 root operator 5979 Nov 5 10:00 .ellepro.e
-rw-r--r-- 1 root operator 44 Nov 5 10:00 .exrc
-rw-r--r-- 1 root operator 537384 Nov 18 15:20 .fonts.cache-1
-rw-r--r-- 1 root operator 304 Nov 5 10:00 .profile
drwx----- 2 root operator 192 Nov 25 11:47 .ssh
226 Transfer finished successfully. 0.52 KB/s
ftp> quit
221 Service closing, don't be a stranger.
```

На рис.4 показано обращение к запущенному нами серверу FTP со стороны другого GUI FTP клиента, запущенного на Linux хосте — gFTP.

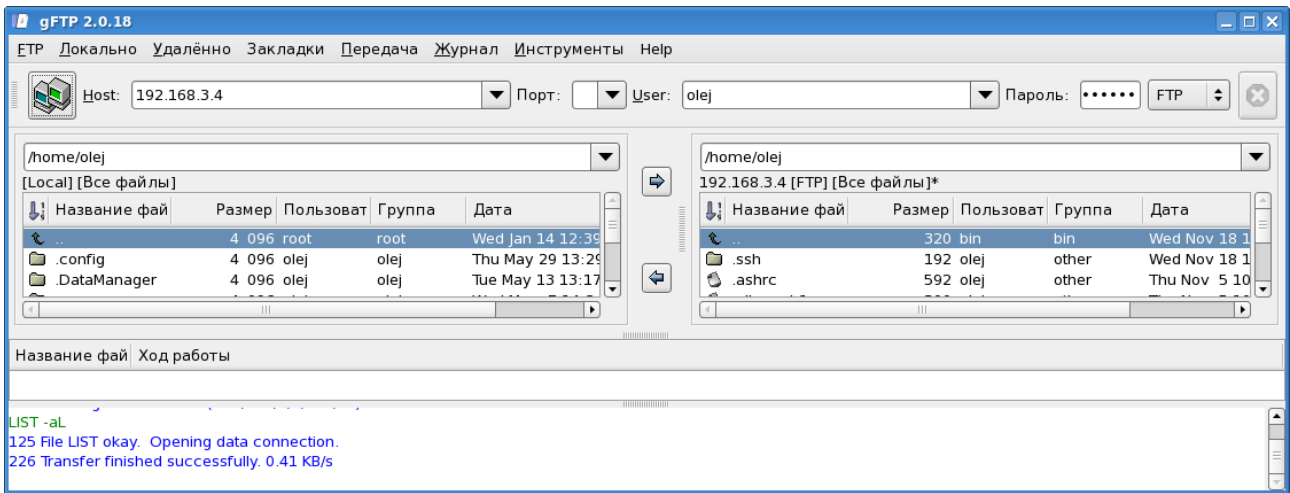


Рис.4

Сервер FTP в MINIX3 — одно из самых ценных приобретений, которые вы можете себе позволить: тем, что он позволяет вам открыть сессию FTP в панели mc в Linux (рис.5, рис.6)

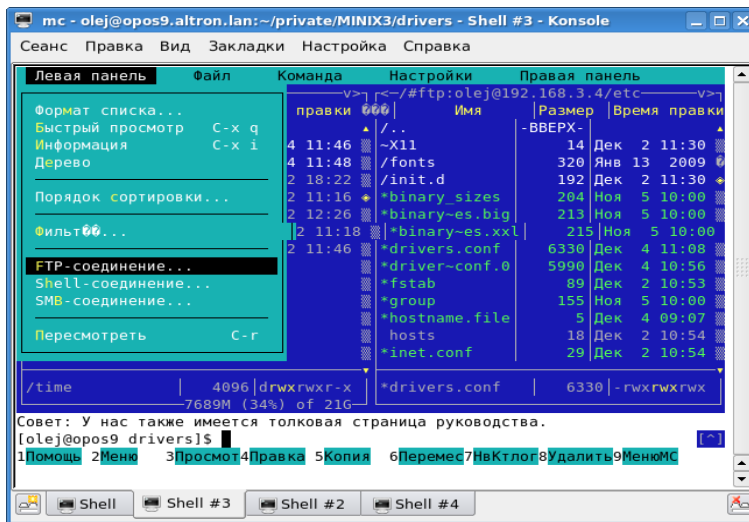


Рис.5

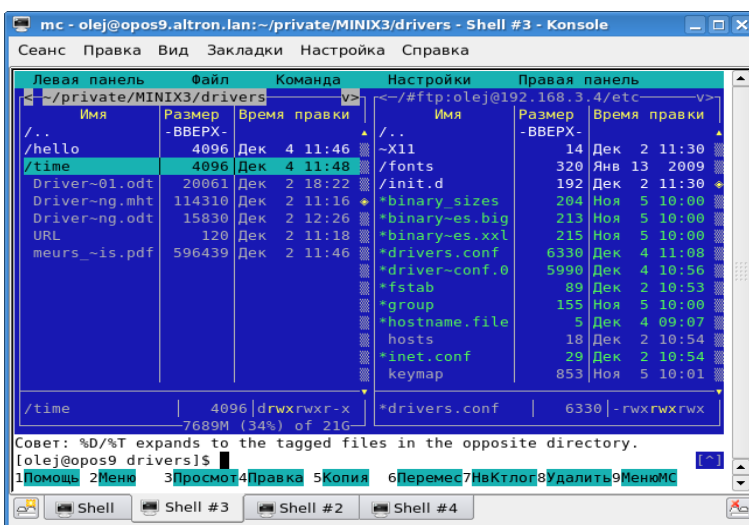


Рис.6

А далее вы сможете копировать и перемещать файлы меж хостами (F5 и F6), и редактировать (F4) файлы MINIX3 хоста, не покидая Linux, и привычными для Linux средствами (рис.7).

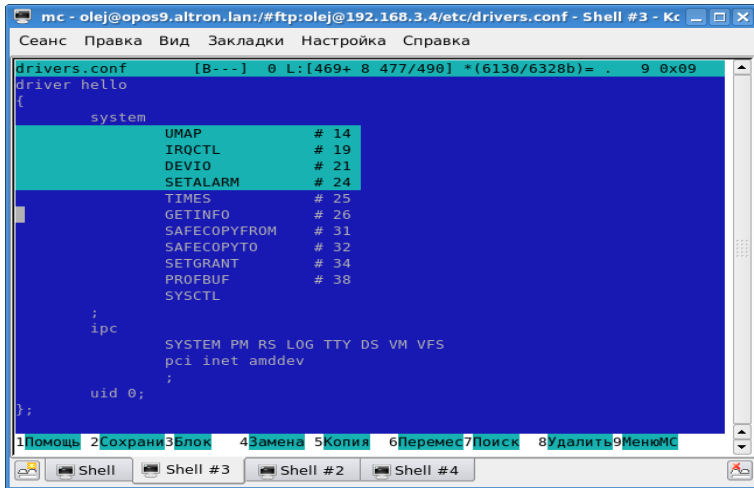


Рис.7

HTTP

Клиент

В MINIX заявлено несколько текстовых HTTP браузеров. Вот один из них (links):

http://www.minix3.ru/files/links_0.99.tar.bz2

Он же представлен на LiveCD в другой версии: #55 links-2.1pre26

Сервер

Сервер представлен в том же виде, что и сервера TELNET и FTP: изначально установленный, исходный код, требующий сборки, представлены даже 2 различающиеся версии:

```
# cd /usr/src/commands
```

```
# ls http*
```

```
httpd:
```

```

Makefile  cgiexec.c  dir2html.sh  httpd.c      httpd0993.txt  pass.h      reply.c
README    config.c   http.h       httpd.conf   net.c          police.c    request.c
SECURITY  config.h   http_status.5 httpd.conf.5 net.h          process.c   utility.c
build     dir2html.c httpd.8      httpd.mtype  pass.c         proxy.c     utility.h

```

```
httpd0995:
```

```

Makefile  config.c   http.h       httpd.conf   net.c          police.c    request.c
README    config.h   http_status.5 httpd.conf.5 net.h          process.c   utility.c
SECURITY  dir2html.c httpd.8      httpd.mtype  pass.c         proxy.c     utility.h
cgiexec.c dir2html.sh httpd.c      httpd0995.txt pass.h         reply.c

```

```
# cd httpd
```

```
# cat README
```

```
...
```

```
COMPILING:
```

To compile httpd all you need to do is type "make" in the httpd source directory. There should be no errors or warnings. If you should run out of memory when compiling try adding the -m option to the CFLAGS list in the Makefile.

```
INSTALLING:
```

To install httpd all you need to do is type "make install" in the httpd

source directory. By default the place to install httpd is into /usr/local/bin. If you would like to change this then change BINDIR in the Makefile. Httpd will be linked to in.httpd, which is the preferred name for a program started by the tcpd internet access control program. The program dir2html is also installed -- this provides a directory listing when a web client accesses a directory which does not contain a file named index.html (or an alternative designated in /etc/httpd.conf). The man pages are installed by typing "make installman".

CONFIGURING:

Before running httpd it must be configured. The name of the default configuration file is /etc/httpd.conf or you may pass the configuration file name to httpd. Upon starting up, httpd will parse the configuration file and then process requests. This README file and the sample httpd.conf may also help in configuring. The httpd.conf.5 man page presents the same information for reference use.

STARTING:

First of all httpd is a server and therefore you will need to start it with tcpd. Tcpd is a program which listens for incoming TCP connections on the passed port and when a connection comes in it forks and starts the given daemon program. Therefore to start httpd you use:

```
tcpd http /usr/local/bin/in.httpd &
```

You will more than likely have this line in your /etc/rc or /etc/rc.net file so that whenever your system is restarted the web server will also be started. The first parameter http is the port that tcpd is going to be listening for connections on. Here http (which should be defined in /etc/services as 80) is the standard port for a web server. The second parameter is the program that tcpd will fork and exec when a connection comes in. The program will then have its stdin and stderr connected to the client. Then the web server program will start running with the tcpd program waiting for the next connection. Currently there is no ability to limit the number of simultaneous web servers running. NOTE: At some point I will be adding the ability for httpd to start itself without the need of tcpd. That way httpd will already be in memory and have parsed its configuration file.

In Minix 2.0.3 and later versions you may use:

```
daemonize tcpd http /usr/local/bin/in.httpd
```

(daemonize is a shell function defined in /usr/etc/rc which starts programs as daemons).

...

Собираем сервер:

```
# make
```

```
cc -c -O -D_MINIX -D_POSIX_SOURCE -DDAEMON=1 httpd.c
```

```
cc -c -O -D_MINIX -D_POSIX_SOURCE -DDAEMON=1 utility.c
```

```

cc -c -O -D_MINIX -D_POSIX_SOURCE -DDAEMON=1 request.c
cc -c -O -D_MINIX -D_POSIX_SOURCE -DDAEMON=1 process.c
cc -c -O -D_MINIX -D_POSIX_SOURCE -DDAEMON=1 reply.c
cc -c -O -D_MINIX -D_POSIX_SOURCE -DDAEMON=1 police.c
cc -c -O -D_MINIX -D_POSIX_SOURCE -DDAEMON=1 cgiexec.c
cc -c -O -D_MINIX -D_POSIX_SOURCE -DDAEMON=1 net.c
cc -c -O -D_MINIX -D_POSIX_SOURCE -DDAEMON=1 config.c
cc -c -O -D_MINIX -D_POSIX_SOURCE -DDAEMON=1 pass.c
cc -c -O -D_MINIX -D_POSIX_SOURCE -DDAEMON=1 proxy.c
cc -i -o httpd httpd.o utility.o request.o process.o reply.o \
    police.o cgiexec.o net.o config.o pass.o proxy.o
install -S 16kw httpd
cc -c -O -D_MINIX -D_POSIX_SOURCE -DDAEMON=1 dir2html.c
cc -i -o dir2html dir2html.o
install -S 8kw dir2html
# make install
install -cs -o bin httpd /usr/local/bin/httpd
install -l /usr/local/bin/httpd /usr/local/bin/in.httpd
install -cs -o bin dir2html /usr/local/bin/dir2html
# make installman
mkdir -p /usr/local/man/man5
mkdir -p /usr/local/man/man8
cp -p httpd.conf.5 http_status.5 /usr/local/man/man5
cp -p httpd.8 /usr/local/man/man8
makewhatis /usr/local/man

```

Запуск, как и прочих серверов:

```

# tcpd http /usr/local/bin/in.httpd
httpd: Could not read /etc/httpd.conf config file.
httpd: Error reading configuration file.

```

Это говорит нам о том, что HTTP-сервер требует достаточно обстоятельного конфигурирования и наличия определённых конфигурационных файлов. Начальные образцы таких конфигурационных файлов находятся в каталоге исходных кодов сервера:

```

# cp /usr/src/commands/httpd/httpd.conf /etc
# mkdir /usr/www
# mkdir /usr/www/etc
# cp /usr/src/commands/httpd/httpd.mtype /usr/www/etc/httpd.mtype

```

Дальнейшую информацию по настройке и запуску сервера (которые не входят в цель нашего экскурса) почерпните из цитировавшегося файла README и справочной страницы:

```
# man httpd.conf
```

Дальнейшую информацию по настройке и запуску сервера (которые не входят в цель нашего экскурса)

Ещё одна доступная в MINIX3 реализация HTTP сервера — это общеизвестный сервер Apache, портированный под MINIX3. Установка и настройка сервера Apache никак не отличается в MINIX3 от прекрасно описанных

процедур для всех вообще POSIX-систем. Единственно, что необходимо помнить при портировании и установке Apache в MINIX3, это то, что Apache в последних версиях, как правило, собирается или устанавливается и конфигурируется с поддержкой программных потоков (pthread). В MINIX3 отсутствует реализация программных потоков, поэтому сервер Apache должен устанавливаться как «fork-ающий сервер», создающий копию-процесс на каждое подключение пользователя.

Протоколы точка-точка

Протокол TCP/IP не обязательно требует наличия LAN среды. Вполне допустимы другие физические носители, например: последовательные линии с подключением через порты RS-232, линии с подключением через параллельные порты Centronics, скоростные синхронные порты RS-485, высокоскоростные каналы E1/T1 или E3/T3. Эти возможности обеспечиваются специализированными реализациями протокола IP, ниже рассматриваются только некоторые из них.

Особенности последовательного канала в MINIX3

Суть в том, что большинство систем (MS DOS, Windows, Linux, QNX) поддерживают обмен через нуль модемный кабель, который известен как 3-х проводное соединение. MINIX3 **не обеспечивает** канал через 3-х проводное соединение. Я не знаю, работает ли MINIX3 с кабелем 5-ти проводного соединения, но все дальнейшие (и успешные) настройки проводились **только** для соединения RS-232 портов кабелем с 7-ми проводным соединением. На 3-х проводной линии терминальная система будет нормально **принимать** байтовый поток с RS-232, но при **передаче** будет возвращать управление так, как будто операция выполнена, фактически ничего не передавая.

Электрическая схема 7-ми проводного нуль модемного кабеля приводится в приложении.

SLIP

Первым рассмотрим из протоколов на последовательной линии: Serial Line IP — это самая первая реализация TCP/IP «для бедных», широко используется до сих пор для подключения через RS-232. Для организации IP канала нам предстоит настроить SLIP соединение с 2-х хостов: Linux (IP адрес 192.168.6.6) и MINIX3 (IP адрес 192.168.6.4).

1. В Linux SLIP запускается примерно так (там может быть ещё множество параметров, которые уточняем в man), показан запуск на порту COM2 (/dev/ttyS1):

```
# slattach -p slip /dev/ttyS1
```

```
....
```

2. Смотрим что изменилось у нас с сетевыми интерфейсами:

```
# ifconfig s10
```

```
s10      Link encap:Serial Line IP
         POINTOPOINT NOARP MULTICAST  MTU:296  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:10
         RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

- появился новый интерфейс s10.

3. Новый сетевой интерфейс (s10) создан, но у него пока ещё нет IP адреса, присвоим ему IP адрес:

```
# ifconfig s10 192.168.6.6
```

```
# ifconfig s10
```

```

sl0      Link encap:Serial Line IP
        inet addr:192.168.6.6  P-t-P:192.168.6.6  Mask:255.255.255.255
        UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:296  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:10
        RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

```

4. Но создание сетевого ещё мало — нам нужно обеспечить отправку любых пакетов для посети этого интерфейса 192.168.6.* (роутинг) именно через интерфейс sl0, для этого правим таблицу роутинга (добавляем запись подсети):

```
# route -v add -net 192.168.6.0 netmask 255.255.255.0 gw 192.168.6.6
```

```
# route
```

```
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.1.0	*	255.255.255.248	U	0	0	0	eth0
192.168.6.0	192.168.6.6	255.255.255.0	UG	0	0	0	sl0
169.254.0.0	*	255.255.0.0	U	0	0	0	eth0
default	192.168.1.1	0.0.0.0	UG	0	0	0	eth0

К этому моменту мы подготовил канал SLIP со стороны Linux, но нам ещё необходим встречный канал со стороны MINIX3.

1. Запускаем SLIP интерфейс и привязываем его к последовательному каналу (COM1) /dev/tty00:

```
# term /dev/tty00
```

```
<Ctrl>+<J><S>
```

```
# slip /dev/psip1 <&9 >&9
```

Примечание: для справки - сетевые интерфейсы при старте MINIX3 хоста были определены в файле /etc/inet.conf как:

```
# cat /etc/inet.conf
```

```
eth0 dp8390 0 ;
psip1 { default; } ;
```

Нас здесь интересует интерфейс /dev/psip1, он же /dev/ip1 :

```
# ls /dev/*ip*
```

```
/dev/ip  /dev/ip0  /dev/ip1  /dev/ipstat  /dev/psip  /dev/psip1
```

2. Связываем сетевой интерфейс (-I) /dev/ip1 (/dev/psip1) с IP адресом (адресом интерфейса -h) 192.168.6.4:

```
# ifconfig -I /dev/psip1 -h 192.168.6.4
```

```
# ifconfig -av
```

```
/dev/ip1: address 192.168.6.4 netmask 255.255.255.0 mtu 576
```

3. Добавляем роутинг для этого интерфейса на хост Linux:

```
# add_route -g 192.168.6.4
```

Позже, убедившись что это работает, мы можем вписать эту команду, вместе с предыдущим ifconfig, в

стартовый конфигурационный файл /etc/rc.net.

```
# pr_routes
ent #   if          dest          gateway dist  pref  mtu flags
     0   ip1        0.0.0.0/0     192.168.6.4  1     0    0 static
```

Вот к этому времени у нас IP канал сквозь нуль модемный RS-232 кабель уже установлен. На Linux хосте выполняем:

```
# ping 192.168.6.4
PING 192.168.6.4 (192.168.6.4) 56(84) bytes of data.
64 bytes from 192.168.6.4: icmp_seq=1 ttl=96 time=184 ms
64 bytes from 192.168.6.4: icmp_seq=2 ttl=96 time=183 ms
64 bytes from 192.168.6.4: icmp_seq=3 ttl=96 time=183 ms
--- 192.168.6.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 183.945/184.073/184.326/0.526 ms
```

И далее:

```
# ssh 192.168.6.4
root@192.168.6.4's password:
Last login: Sun Nov 29 11:37:08 2009
```

...

```
# uname -a
Minix 192.168.6.4 3 1.5 i686
```

```
# ping 192.168.6.6
192.168.6.6 is alive
```

Для убедительности (передачи достаточно объёмного файла), в этой же сессии SSH можем выполнить что-то типа следующего:

```
# cd /usr/src/drivers/tty
# ls -l rs232.c
-rw-r--r-- 1 bin operator 26068 Apr  6 2009 rs232.c
```

```
# cat rs232.c
```

...

- все эти операции выполнялись на хосте MINIX3 (хотя сами команды и вводились в SSH сеансе с хоста Linux).

Примечание: полученный IP канал несколько «со странностями», например, со стороны Linux:

```
# traceroute 192.168.6.4
traceroute to 192.168.6.4 (192.168.6.4), 30 hops max, 40 byte packets
send: Недостаточно буферного пространства
```

Однако его вполне достаточно для подключения сетевых клиентов и выполнения пользовательских операций, таких как передача файлов, например, что и было продемонстрировано в листингах выше.

В принципе, man рекомендует несколько другой способ запуска slip, воспользуемся ним:

```
# ( stty raw ; slip /dev/psip1 ) </dev/tty00 >/dev/tty00
```

...

Вот, достаточно показательный, как мне кажется, выполненный после такого подключения сеанс SSH, «туда и обратно»:

```
# ssh 192.168.6.4
root@192.168.6.4's password:
Last login: Sun Nov 29 16:16:38 2009 from 192.168.6.6
...
# uname
Minix
# ssh 192.168.6.6
root@192.168.6.6's password:
Last login: Sun Nov 29 18:12:03 2009 from 192.168.6.4
# uname
Linux
...
# exit
Connection to 192.168.6.6 closed.
#
```

CSLIP

Compressed SLIP - модификация протокола SLIP. Сжатию подвергаются заголовки IP-пакетов. Но не сами данные!

Примечание: В стандартном IP-пакете IP-заголовки со служебной информацией занимают порядка 40 байт. В CSLIP эти заголовки сжимаются до 3 байт. При пересылке большого числа мелких пакетов получается существенный выигрыш в скорости. Если пакеты крупные, то выигрыша не будет никакого.

PPP

Point-to-Point Protocol — более поздний по сравнению со SLIP, и более широко известный и используемый способ инкапсуляции последовательного потока в IP.

Примечание: PPP для совместимости поддерживает режимы эмуляции SLIP и CSLIP.

Работы по PPP в MINIX2 велись (в версии 2.0.4), текущее их состояние я оценить не готов.

PLIP

Parallel Line IP — инкапсуляция в IP потока данных через соединение компьютеров посредством Centronix параллельных портов. Может казаться экзотикой, но на удивление часто используется для межмашинной связи в мире Linux. Достоинство: достаточно высокая физическая скорость канала (в сравнении с RS-232), описывается [7], что скорость соединения (уже на уровне IP канала) составляет от 5 кБ/сек до 40 кБ/сек. Требует изготовления несложного специального кабеля, кабель детально описан в [7], и во многих других источниках. В man-ax MINIX3 достаточно много внимания уделяется PLIP, судя по описаниям, он проработан не менее, чем SLIP. Больше я ничего не могу добавить относительно PLIP — для проверки его возможностей требуется изготовить специальный кабель, но уже изложенной здесь информации вполне достаточно, чтобы проделать все необходимые шаги для установления PLIP канала.

Инструменты сети

Рассматриваемые далее вопросы не являются напрямую рассмотрением конкретных сетевых протоколов MINIX3, но оказываются необходимыми к рассмотрению при любой работе с сетью.

Конфигурирование сети MINIX3

Здесь мы приведём, по итогам уже проведенного изучения, пример возможного вида конфигурационных файлов MINIX3, изменённых и дополненных так, чтобы основной набор сетевых средств стартовал непосредственно с загрузки системы (изменённые или добавленные строки показаны *жирным курсивом*):

```

# cat /etc/profile
RC_TZ=/etc/rc.timezone
export TZ=GMT0
if [ -f "$RC_TZ" ]
then . "$RC_TZ"
fi
export MANPATH=/usr/man:/usr/local/man:/usr/gnu/man:/usr/X11R6/man
tcpd telnet /usr/bin/in.telnetd &
tcpd ftp /usr/bin/in.ftpd &
tcpd login /usr/bin/in.rlogind &

# cat /etc/inet.conf
eth0 dp8390 0 { default; } ;
psip1 ;
psip2 ;

# cat /etc/ttytab | head -n 10
# ttytab - terminals
#
# Device          Type          Program      Init
console          minix        getty
ttyc1            minix        getty
ttyc2            minix        getty
ttyc3            minix        getty
#tty00           unknown
#tty01           unknown
ttyp0            network

```

Утилиты сети

Здесь собрана краткая сводка по сетевым утилитам, как они реализованы в MINIX3. Эти программы предназначены не для осуществления сетевых коммуникаций, но для диагностики и устранения неисправностей в сети. Набор сетевых утилит (и структура сетевого стека) MINIX3 очень значительно отличается от общепривычного сетевого стека, как он представлен в Linux, SunSolaris, QNX и других операционных системах. Поэтому включение в рассмотрение специфических сетевых утилит MINIX3 никак не кажется лишним.

ping

ping, в общем, общеизвестная программа для проверки доступности хоста (прохождения протокола ICMP).

Со стороны Linux:

```

# ping 192.168.2.4
PING 192.168.2.4 (192.168.2.4) 56(84) bytes of data.
64 bytes from 192.168.2.4: icmp_seq=1 ttl=128 time=1.45 ms
64 bytes from 192.168.2.4: icmp_seq=2 ttl=128 time=1.30 ms
64 bytes from 192.168.2.4: icmp_seq=3 ttl=128 time=1.30 ms
--- 192.168.2.4 ping statistics ---

```

```
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 1.301/1.352/1.453/0.082 ms
```

Со стороны MINIX3 совсем немногословный:

```
# ping 192.168.3.6
192.168.3.6 is alive
```

```
# man ping
man: no manual on ping
```

ifconfig

Так в Linux выглядит интерфейс виртуальной сети к MINIX3 когда он выполняется в системе QEMU:

```
# ifconfig tap0
tap0      Link encap:Ethernet  HWaddr 5A:D1:C5:12:E6:C9
          inet addr:192.168.3.6  Bcast:192.168.3.255  Mask:255.255.255.0
          inet6 addr: fe80::58d1:c5ff:fe12:e6c9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4625 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12182 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:427912 (417.8 KiB)  TX bytes:1825693 (1.7 MiB)
```

В MINIX3 ifconfig выглядит совершенно иначе:

```
# ifconfig -av
/dev/ip0: address 192.168.3.4 netmask 255.255.255.0 mtu 1500
```

В обеих системах ifconfig используется (и мы его вынуждены использовать интенсивно) для присвоения адреса IP сетевому интерфейсу. При этом в MINIX3 ifconfig использует совершенно непривычный синтаксис:

```
# ifconfig -I /dev/ip1 -h 192.168.3.4
```

tcpstat

Утилита диагностирует текущие активные соединения:

```
# tcpstat
 2 qemu-minix:ssh <- 192.168.3.6:53312 ESTABLISHED
    RQ: 0, SQ: 0, RWnd: 32768, SWnd: 1460, SWThresh: 32768
 3 qemu-minix:ssh <- 192.168.3.6:35738 ESTABLISHED
    RQ: 0, SQ: 0, RWnd: 32768, SWnd: 1460, SWThresh: 32768
```

Информация по утилите ограничена:

```
# man tcpstat
man: no manual on tcpstat
# tcpstat -h
illegal option -- h
```


Usage: tcpstat [-anv]

add_route

Ключевая команда в настройке сети MINIX3 — добавление записи маршрутизации. Примеры:

```
# add_route -g 192.168.3.6
# add_route -g 192.168.3.6 -d 192.168.3.0 -n 255.255.255.0
```

Комплементарная ей команда: del_route — удаление маршрута из таблицы.

pr_routes

Вывод текущего содержимого таблицы маршрутизации:

```
# pr_routes -a
```

Как выйти во внешнюю сеть?

После установки MINIX3 и настройки сети, мы, первейшим делом, должны проверить достижимость сетевых хостов. Ниже показаны примеры установленного под QEMU (в реальной установке всё аналогично, но проще) MINIX3 (v.3.1.6) с сетевым интерфейсом:

```
# ifconfig -av
/dev/ip0: address 192.168.3.7 netmask 255.255.255.0 mtu 1500
```

В этой же подсети создан виртуальный интерфейс QEMU в Linux:

```
# ifconfig tap0
tap0      Link encap:Ethernet  HWaddr DA:D5:FE:A1:AF:66
          inet addr:192.168.3.6  Bcast:192.168.3.255  Mask:255.255.255.0
          inet6 addr: fe80::d8d5:feff:feal:af66/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:56 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:120 (120.0 b)  TX bytes:15955 (15.5 KiB)
```

Проверяем доступность хоста Linux из MINIX3 (а заодно и localhost MINIX3):

```
# ping 192.168.3.7
192.168.3.7 is alive
# ping 192.168.3.6
192.168.3.6 is alive
```

Но:

```
# ping 192.168.1.1
no answer from 192.168.1.1
```

- в показанной конфигурации 192.168.1.1 — это ADSL-шлюз из реальной LAN, в которой находится Linux хост. Как легко видеть, шлюз недостижим (а значит и любые хосты и внутри LAN, и, тем более, снаружи неё) из MINIX3. У нас недостаёт роутинга по умолчанию в реальную LAN и далее (уже средствами самой LAN) во внешние сети. Добавим недостающее:

```
# add_route -g 192.168.3.7 -d 0.0.0.0 -m 1 -n 0.0.0.0 -v
adding output route to 0.0.0.0 with netmask 0.0.0.0 using gateway 192.168.3.7
```

```
# pr_routes -a
ent #   if          dest          gateway dist  pref  mtu flags
     1  ip0          0.0.0.0/0     192.168.3.7   1     0    0 static
# ping 192.168.1.1
192.168.1.1 is alive
```

Вот теперь нам из MINIX3 достижимы все хосты LAN. Убедимся в достижимости и всех внешних хостов Интернет. Выберем (в Linux) любой из известных нам WWW-хостов:

```
$ nslookup qnx.org.ru
Server:          195.5.46.12
Address:         195.5.46.12#53
Non-authoritative answer:
Name:   qnx.org.ru
Address: 72.249.144.181
$ ping 72.249.144.181
PING 72.249.144.181 (72.249.144.181) 56(84) bytes of data.
64 bytes from 72.249.144.181: icmp_seq=1 ttl=48 time=196 ms
64 bytes from 72.249.144.181: icmp_seq=2 ttl=48 time=196 ms
--- 72.249.144.181 ping statistics ---
3 packets transmitted, 2 received, 33% packet loss, time 2000ms
rtt min/avg/max/mdev = 196.898/196.931/196.964/0.033 ms
```

А теперь то же самое из MINIX3:

```
# ping 72.249.144.181
72.249.144.181 is alive
```

Разрешение сетевых имён

Судя по всему, для MINIX3 пока не реализованы какие либо средства разрешения сетевых имен (типа nslookup — обращения к DNS). Единственный доступный механизм разрешения — через записи, прописанные в файле /etc/hosts; такое разрешение имён осуществляет специфический для MINIX3 демон nonamed. Рассмотрим:

```
# cat /etc/hosts
192.168.3.7      %nameserver   #minix
64.102.255.44   %nameserver   #DNS 1
128.107.241.185 %nameserver   #DNS 2
192.168.3.7     minix
```

- где имена DNS-серверов (64.102.255.44 и 128.107.241.185) прописаны в ходе начального диалога инсталляции системы (и, похоже, единственной целью вопросов о DNS при инсталляции и есть запись их в /etc/hosts — никакого иного действия они не оказывают). После установления роутинга, как показано выше:

```
# ping 64.102.255.44
64.102.255.44 is alive
# ping 128.107.241.185
128.107.241.185 is alive
```

- сами хосты DNS-серверов, как мы видим, доступны, но:

```
# ping qnx.org.ru
ping: unknown host (qnx.org.ru)
```

```
# ftp qnx.org.ru
Unresolved host qnx.org.ru
...
```

И если мы добавим в строку:

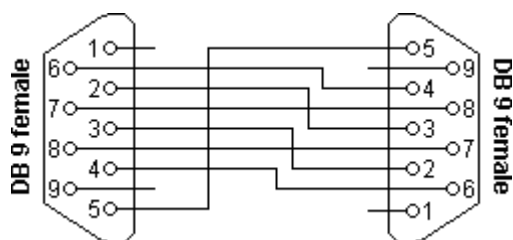
```
72.249.144.181 qnx.org.ru
```

то

```
# ping qnx.org.ru
```

```
qnx.org.ru is alive
```

Приложение: Кабель для SLIP соединения компьютеров



На схеме приведен кабель для полного 7-ми проводного соединения по последовательному каналу RS-232 2-х компьютеров.

Дополнительные источники информации

Ссылки на man я даю по размещённым в Интернет файлам для версии MINIX v.2.0.4. Они незначительно отличаются от версии 3; в работающей системе все man можно получить непосредственно в консоли системы (или в терминал удалённого доступа TELNET, RLOGIN, SSH) по имени команды.

1. man страница по ftpd:

<http://minix1.woodhull.com/current/2.0.4/wwwman/man8/in.ftpd.8.html>

2. man страница по tcpd:

<http://minix1.woodhull.com/current/2.0.4/wwwman/man8/tcpd.8.html>

3. man страница по rlogind:

<http://minix1.woodhull.com/current/2.0.4/wwwman/man8/in.rld.8.html>

4. man страница по httpd:

<http://minix1.woodhull.com/current/2.0.4/wwwman/man8/in.httprd.8.html>

- детальное описание достаточно громоздкой настройки httpd.

5. man страница по rshd:

<http://minix1.woodhull.com/current/2.0.4/wwwman/man8/rshd.8.html>

6. Организация TCP/IP по последовательным линиям

<http://lib.ru/unixhelp/slip.txt>

7. LINUX PLIP MINI-HOWTO

http://ruslandh.narod.ru/howto_ru/mini/PLIP/

<http://rus-linux.net/MyLDP/MINI-HOWTO-ru/PLIP.html>

- перевод на русский, полностью описывающий PLIP, начиная с кабеля, и заканчивая использованием в гетерогенных соединениях, меж различными ОС.

8. man serial-ip

<http://minix1.woodhull.com/current/2.0.4/wwwman/man8/serial-ip.8.html>

- такой команды или файла нет, это общие рассуждения о том, как поднять IP над последовательной линией.

9. man ip

<http://minix1.woodhull.com/current/2.0.4/wwwman/man4/ip.4.html>

- описание на уровне программного кода, но очень проясняющее понятия сетевого интерфейса, и описывающее аварийные коды завершения, которые можно наблюдать при запуске сетевых программ в неправильных конфигурациях.

10. man страница по команде add_route:

http://minix1.woodhull.com/current/2.0.4/wwwman/man8/add_route.8.html

- управление маршрутизацией IP.

11. man страница по команде pr_routes:

http://minix1.woodhull.com/current/2.0.4/wwwman/man8/pr_routes.8.html

- индикация таблицы маршрутизации.