Мультипротокольные маршрутизаторы NSG

Программное обеспечение NSG Linux

Руководство пользователя

Часть 6

Основные команды и утилиты NSG Linux

Версия программного обеспечения 1.0 build 3

Обновлено 23.06.2008

Москва 2008

АННОТАЦИЯ

Данный документ содержит руководство по настройке и применению мультипротокольных маршрутизаторов NSG, оснащенных программным обеспечением NSG Linux. Руководства по применению других продуктов NSG, а также базового программного обеспечения NSG для серий NPS–7e, NSG–500, NX–300 и NSG–800 содержатся в отдельных документах.

Документ состоит из следующих разделов:

Часть 1. Общесистемная конфигурация.
Часть 2. Физические порты.
Часть 3. Протоколы канального уровня. Коммутация пакетов.
Часть 4. Маршрутизация и службы IP.
Часть 5. Туннелирование и виртуальные частные сети (VPN).
Часть 6. Основные команды и утилиты NSG Linux.

В шестой части документа изложены начала работы с ОС Linux в объёме, желательном для администрирования и отладки сетей на основе оборудования NSG с использованием расширенных возможностей системы. Документ рассчитан на пользователей, имеющих представление об управлении ПК при помощи командной строки, но не знакомых с ОС Linux.

Кроме того, описаны некоторые системные утилиты и их применение. Эта часть документа предназначена для пользователей, знакомых с ОС Linux и желающих максимально полно использовать её возможности вне рамок основной командной оболочки NSG.

Общее описание системы, описание общесистемных параметров и командного языка системы приведены в Части 1. Настройка физических интерфейсов различного типа представлена в Части 2. Настройка протоколов канального уровня (в т.ч. VLAN, организация сеансового доступа средствами PPP и доступа к асинхронным портам средствами Reverse Telnet), коммутация пакетов на втором уровне (Ethernet bridging, Frame Relay) и коммутация пакетов X.25 рассмотрены в Части 3. Настройка IP-маршрутизации и связанных с ней служб, а также механизмов управления IP-трафиком и обеспечения QoS, описана в Части 4. Часть 5 посвящена построению туннелей и виртуальных частных сетей различных типов.

ВНИМАНИЕ Продукция компании непрерывно совершенствуется, в связи с чем возможны изменения отдельных аппаратных и программных характеристик по сравнению с настоящим описанием. Сведения о последних изменениях приведены в файлах README.TXT, CHANGES, а также в документации на отдельные устройства.

Замечания и комментарии по документации NSG принимаются по адресу: doc@nsg.net.ru.

ООО «Эн–Эс–Джи»
Россия 105187 Москва
ул. Кирпичная, д.39, офис 1302
Тел.: (+7–495) 918–32–11
Факс: (+7–495) 918–27–39

http://www.nsg.ru/
mailto:info@nsg.net.ru
mailto:sales@nsg.net.ru
mailto:support@nsg.net.ru

§ СОДЕРЖАНИЕ §

Часть 6. Основные команды и утилиты NSG Linux.

§6.1. Основы работы в ОС Linux

§6.1.1. Документация по ОС Linux

Ввиду особенностей развития ОС Linux как распределенного проекта, неподконтрольного какому-либо одному центру или организации, для данной системы нет и не может быть единого исчерпывающего руководства, которое можно было бы загрузить в память пользователя. Работа с Linux предполагает непрерывное саморазвитие пользователя и самостоятельное освоение тех компонент системы, которые становятся актуальными для него по мере решения очередных задач. Более или менее едиными являются только форматы написания руководств и стиль командного языка.

Большинство команд и утилит ОС Linux документировано в виде так называемых страниц руководств — *man pages*. Они входят в состав практически всех дистрибутивов Linux, ориентированных на конечного пользователя. Для просмотра *man pages* следует войти в текстовый терминал и ввести команду

man *имя_команды/утилиты*

Для пользователей, работающих в других операционных системах, *man pages* доступны на многих интернет-ресурсах, посвящённых Linux, например:

http://www.opennet.ru/man.shtml

Краткую встроенную подсказку по формату команды и её опций командной строки можно просмотреть, как правило, вызвав команду с опцией --help, -h или без каких-либо опций и параметров. В отличие от весьма объёмистых *man pages*, встроенная подсказка в большинстве случаев присутствует и в специализированных сборках Linux, таких как *embedded Linux* для различных устройств и, в частности, система NSG Linux:

```
# tftp --help
BusyBox v1.4.1 (2008-04-23 19:06:31 MSD) multi-call binary
Usage: tftp [OPTION]... HOST [PORT]
Transfer a file from/to tftp server using "octet" mode
Options:
    –l FILE    Local FILE
    –r FILE    Remote FILE
    –g         Get file
    –p         Put file
    –b SIZE    Transfer blocks of SIZE octets
```

ПРИМЕЧАНИЕ Версии команд и утилит, используемые в различных сборках Linux, могут и будут отличаться. По этой причине, если нет исчерпывающей документации по конкретной системе, рекомендуется ознакомиться с командой в целом по любым доступным *man pages*, а затем руководствоваться встроенным *help* в используемой реализации.

Данная часть Руководства пользователя NSG Linux не имеет своей целью полноценное обучение работе в ОС Linux. Её первый раздел следует рассматривать исключительно как краткий справочник по командам, которые могут быть актуальны для сетевого администратора устройств NSG — особенно в случаях диагностики и отладки проблемных ситуаций, снятия расширенной системной информации для отправки в службу технической поддержки NSG. Последующие разделы данной части посвящены специфическим возможностям системы NSG Linux, выходящим за рамки основной командной оболочки. В заключение описаны фирменные и доработанные утилиты NSG, используемые для реализации отдельных функций, и их вызов непосредственно из командной оболочки Linux.

§6.1.2. Вход в систему и переключение между облочками

Для работы в командной оболочке ОС Linux (ash) следует войти в систему под именем root. Доступ возможен любыми средствами, предусмотренными для работы с устройством в режиме командной строки: через консольный порт, Telnet, SSH, PAD. После входа системное приглашение принимает вид:

```
#
```

ПРИМЕЧАНИЕ Пароли для системных пользователей nsg и root устанавливаются средствами основной командной оболочки (см. Часть 1).

Для эпизодических действий можно вызвать командную оболочку Linux из меню enable основной оболочки при помощи команды:

```
start-shell
```

Наоборот, из командной облочки Linux всегда можно вызвать основную оболочку при помощи команды:

vtysh

Для выхода из вложенной командной оболочки в исходную необходимо, в обоих случаях, выполнить команду:

exit

Если эта команда выполняется в первоначальной оболочке, то пользователь выходит из системы.

Во многих случаях бывает удобно иметь одновременно два сеанса работы с системой — один как nsg, другой как root. Например, подключиться к устройству через два сеанса Telnet, или через Telnet и через консольный порт. При этом основная командная оболочка допускает одновременнную работу двух пользователей с именем nsg, но только один из них может работать в режиме enable. Однако оболочка Linux может быть запущена независимо от неё и не испытывает каких-либо ограничений на изменение конфигурации устройства. В частности, с её помощью можно завершить сеанс работы основной оболочки, если она функционирует неадекватно.

ПРИМЕЧАНИЕ Командная оболчка Linux в целом, и утилита config-nsg в частности, не проверяют, работает ли в это время другой пользователь в режиме enable. Таким образом, при использовании командной оболочки Linux может возникнуть ситуация, когда устройство настраивается двумя администраторами одноврменно.

§6.1.3. Командная строка

При вводе команд Linux можно перемещаться по вводимой командной строке с помощью клавиш ← и →, редактировать сожержимое строки с помощью клавиши Backspace. Использовать клавишу Delete не рекомендуется, поскольку производимое ею действие может варьироваться в зависимости от настроек терминальной программы.

Можно использовать редактор команд, позволяющий повторять ранее введённые команды с помощью клавиш ↑ и ↓.

При вводе директорий и файлов команд можно набирать только начальную часть, идентифицирующую их полностью, а затем использовать клавишу TAB для автодополнения.

§6.1.4. Особенности командного языка *nix для пользователей других ОС

Пользователям, привыкшим к соглашениям, синтаксису языка и структуре файловой системы, принятым в других операционных системах, следует обратить особое внимание на следующие отличия, характерные для всех Unix-подобных систем:

1. Файловая структура всегда представляется единым деревом, имеющим один корень, независимо от количества физических накопителей и разделов на них. В это же дерево монтируются, при необходимости, сетевые диски и сменные носители.
2. Для обозначения перехода из одной директории в другую всегда используется символ "прямой слэш" (/), и только он. Использование обратного слэша в этом смысле не допускается. В частности, широко используются следующие обозначения:
   - / Корневая директория файловой системы на данном устройстве
   - ./ Текущая директория
   - ../ Родительская директория
3. Если путь, указанный в параметрах вызова какой-либо команды, начинается с /, то он отсчитывается от корня файловой системы. В противном случае путь отсчитывается от текущей директории.
4. Имена файлов и директорий могут иметь произвольную длину, могут содержать точки и обратные слэши, но не могут содержать пробелы, знаки табуляции и символы & ; ( ) > < | .
5. Механизм ассоциаций между расширениями (суффиксами) файлов и приложениями, которыми следует открывать эти файлы, используется, как правило, только в рамках графических оболочек. В общем случае часть имени файла, расположенная после крайней правой точки, никакого специального смысла для системы не несёт, а имя может содержать произвольное число точек, или не содержать их вовсе. В некоторых случаях, наоборот, принято использовать двойной суффикс (например, myfile.tar.gz).
6. Большие и маленькие буквы во всех случаях различаются. (Исключения из этого правила, например, имена Web-ресурсов, оговариваются особо).
7. Для каждого файла и директории устанавливаются определённые права доступа, в т.ч. на исполнение данного файла. В частности, эти права имеют силу и для текстовых файлов, поскольку любой текстовый файл может рассматриваться как сценарий, или *скрипт*.

§6.1.5. Базовые файловые операции

Для выполнения основных операций с файлами и директориями используются команды:

ls [*путь*] Вывести содержимое директории. В выводимом списке директории обозначаются жирным шрифтом, файлы — обычным.

cd *директория*
chdir *директория*
Перейти в указанную директорию. Обе команды эквивалентны.

mkdir *директория*
Создать директорию с указанным именем.

rm *файл*
Удалить файл с указанным именем.

rm *директория*
Удалить директорию с указанным именем.

cp *источник назначение*
Копировать файл или директорию.

chmod +*права файл*
chmod –*права файл*
Установить и снять, соответственно, права доступа для указанного файла. Некоторые наиболее актуальные права:

r	Чтение
w	Запись
x	Исполнение файла или вход в директорию

Пример: chmod +wx myfile.name

tar –*опции источник/назначение*
Архивировать и разархивировать указанные файлы и директории. Некоторые наиболее актуальные опции:

c	Поместить файлы в архив.
x	Извлечь файлы из архива.
z	При архивировании сжимать и восстанавливать файлы с помощью утилиты gzip; в *nix-системах сжатие и архивирование традиционно считаются двумя независимыми операциями.
j	При архивировании сжимать и восстанавливать файлы с помощью утилиты bzip2.
f *архив*	Имя архива (f ставится последним в пакете опций).

Первым из указанных опций должно стоять c либо x, остальные опции могут комбинироваться с ними. Архив всегда создаётся и восстанавливается с сохранением структуры директорий.
Примеры:

tar –cf /mnt/nfs/nsgconfig.tar /etc
Упаковать всё содержимое директории /etc в файл /mnt/nfs/nsgconfig.tar.gz, без сжатия.

tar –xzf /tmp/nsgconfig.tar.gz
Распаковать архив /tmp/nsgconfig.tar, предварительно сжатый gzip. При этом, поскольку директория-назначение не указана,разархивированная структура директорий будет помещена либо в корневую, либо в текущую директорию, в зависимости от того, как была указана директория-источник при создании архива.

Подробно о синтаксисе и опциях данных команд см. соответствующие *man pages*.

§6.1.6. Стандартные потоки и перенаправление ввода-вывода

Каждая программа при исполнении использует три стандартных потока ввода-вывода: stdin, stdout и stderr, имеющие дескрипторы 0, 1 и 2, соответственно. По умолчанию, все три потока соответствуют консоли, с которой работает пользователь.

При написании скриптов Linux широко используются перенаправления стандартных потоков, в частности:

команда > *файл*
Направить вывод команды в файл, вместо stdout. Если файл с таким именем уже существует, его исходное содержимое будет утрачено.

команда >> *файл*
Направить вывод команды в дополнение к файлу, вместо stdout. Если файл с таким именем уже существует, новый вывод будет дописан в его конец.

команда < *файл*
: Ввести в команды данные из файла, вместо stdin.

команда1 | *команда2*
: Направить вывод команды на вход другой команды, вместо stdout.

n>&*m*
: Перенаправить поток *n* в поток *m*, в частности:
  1>&2 Перенаправить stdout в stderr.
  2>&1 Перенаправить stderr в stdout.

Подробно о перенаправлении потоков ввода-вывода см. страницы руководства по *ash*.

Для того, чтобы избавиться от нежелательного вывода, его обычно направляют в файл с именем /dev/null.

§6.1.7. Просмотр и фильтрация текстовых файлов

Для просмотра текстовых файлов можно использовать следующие команды:

cat *файл* Вывести содержимое файла на консоль (строго говоря — в стандартный поток вывода).

more *файл*
: Вывести содержимое файла на консоль порциями по 21 строке. Для продолжения вывода следует нажать клавишу Enter (на одну строку), Пробел (на один экран) или ↓ (до конца).

less *файл*
: Вывести содержимое файла на консоль с буферизацией. Выведенный текст можно прокручивать на экране с помощью клавиш ↑ и ↓. Команда предоставляет достаточно широкие возможности, в т.ч. перемещение в заданное место файла, поиск заданной подстроки и т.п.

grep *подстрока файл*
: Поиск и вывод строк файла, содержащих указанную подстроку. Вместо подстроки может быть использовано также регулярное выражение, отрицание и т.п. Следующий пример выводит строки системного журнала, относящиеся к процессу pppd под номером 651:
  grep pppd(651) /var/log/messages

Подробно о синтаксисе и опциях данных команд см. соответствующие *man pages*.

§6.1.8. Создание и редактирование текстовых файлов

Для создания и редактирования текстовых файлов (например, файла меню для SMS-управления) в составе NSG Linux имеется текстовый редактор nano. Вызов редактора:

nano *файл*

Редактор работает в экранном режиме, имеет встроенные подсказки и простое меню.

На практике для пользователей, привыкших к возможностям текстовых редакторов в других операционных системах, удобнее всего, как правило, создать файл (как минимум, его первоначальный вариант) на своём компьютере и затем передать его на устройство NSG посредством TFTP (см. п.6.1.9). После этого целесообразно использовать nano для внесения небольших исправлений.

"Ортодоксальный" способ создать текстовый файл — ввести его вручную с консоли, используя перенаправление ввода. Ввод завершается нажатием клавиши Ctrl-D:

```
cat >/etc/nsgsms.conf
    вставить или набрать текст
    Enter
    Ctrl+D
```

§6.1.9. Приём и передача файлов

Для обмена файлами с другими машинами, в частности, для резервирования и восстановления конфигурации устройства, или для установки специфических файлов, не редактируемых средствами основной командной оболочки (например, nsgsms.conf), можно использовать следующие утилиты:

tftp Клиент TFTP

ftpget Клиент FTP для приёма файлов

ftpput Клиент FTP для передачи файлов

wget Клиент HTTP

Чтобы получить подробную справку о ключах и аргументах любой утилиты, введите её имя с ключом --help.

Для обмена файлами с машинами под управлением Unix-систем можно использовать файловую систему NFS и монтировать удалённую директорию в локальную файловую систему, например:

```
mount –t nfs –o nolock 192.168.0.2:/config_backups /mnt/nfs
.....
umount /mnt/nfs
```

В терминах других ОС эта операция называется подключением сетевого диска.

§6.1.10. Системные операции

В отдельных сложных случаях возникает необходимость проконтролировать список исполняемых задач, или даже снять некорректно работающую задачу. Для этих целей используются команды:

ps Вывести список всех процессов. Пример вывода:

```
root@nsg /root # ps
  PID  Uid   VmSize  Stat  Command

    1  root          SW   [swapper]
    2  root          SW   [keventd]
    3  root          SWN  [ksoftirqd_CPU0]
    4  root          SW   [kswapd]
    5  root          SW   [bdflush]
    6  root          SW   [kupdated]
    7  root          SW   [mtdblockd]
    8  root          SW   [nftld]
    9  root          SW   [xot_main]
   10  root          SW   [xot_listener]
   11  root      632 S    init
  151  root          SWN  [jffs2_gcd_mtd6]
  244  root     1296 S    /sbin/zebra -d
  251  root     1528 S    /sbin/ospfd -d
  258  root     2312 S    /sbin/bgpd -d
  265  root     1340 S    /sbin/ripd -d
  277  root          DW   [obj_stat]
  372  root      772 S    /usr/sbin/inetd
  376  root      716 S    -sh
  492  root      664 R    ps
```

Имя процесса, а точнее — командная строка, с которой он был запущен — выводится в последнем столбце. Наиболее существенным, с точки зрения неподготовленного пользователя, является идентификатор процесса (PID). Процессы нумеруются последовательно по мере их запуска. С помощью PID можно установить, например, "завис" ли интересующий пользователя процесс (PID остаётся постоянным), или, наоборот, непрерывно рестартует и тут же аварийно завершается (PID быстро растёт при каждом повторении команды ps). Знание PID также необходимо для выполнения следующей команды:

kill *PID* Завершить процесс с указанным PID. После выполнения команды следует ещё раз ввести команду ps и убедиться, что этот процесс действительно отсутствует в списке.

killall *имя* Завершить все процессы с указанным именем, например, killall pppd завершит все сеансы PPP на данном устройстве. Эту команду удобно также использовать (в т.ч. в скриптах для обработки нештатных ситуаций) для того, чтобы завершить процесс, не выясняя предварительно его PID.

Подробно о синтаксисе и опциях данных команд см. соответствующие *man pages*.

§6.1.11. Перезагрузка устройства

Для перезагрузки устройства средствами командной оболочки Linux следует использовать команду:

```
reboot
```

Для перезагрузки средствами основной командной оболочки следует использовать команду:

```
reload
```

ВНИМАНИЕ Перед перезагрузкой устройства необходимо сохранить текущую конфигурацию и изменения в файловой системе (см.п.6.2.3). В противном случае они будут утрачены.

§6.2. Особенности устройств NSG как Linux-систем

§6.2.1. Структура файловой системы NSG Linux

В файловой системе NSG Linux используются следующие специальные директории для хранения файлов, которые могут потребоваться пользователю:

/etc — Содержит все настройки системы.

/root — Предназначена для хранения файлов пользователя. Де-факто является ссылкой на /etc/root.

/var — Содержит временные файлы и директории, создаваемые системой в ходе работы. Например, вывод системного журнала по умолчанию направляется в файл /var/log/messages.

/tmp — Предназначена для хранения временных файлов пользователя. Например, сюда можно поместить файл конфигурации системы в архивированном виде, передаваемый или принимаемый по tftp. Де-факто является ссылкой на /var/tmp.

Остальные физические директории не предназначены для работы пользователя.

/proc — Виртуальная файловая система, используемая для работы Linux. В NSG Linux, помимо обычных задач, она имеет особое назначение: вся текущая конфигурация устройства, представленная командами основной оболочки, хранится в виде директорий и файлов в директории /proc/sys/nsg. При необходимости они могут быть непосредственно считаны и даже перезаписаны командами и скриптами Linux.

§6.2.2. Хранение файловой системы NSG Linux

Физическое местонахождение и способ хранения файловой системы зависит от типа шасси и наличия энергонезависимой памяти:

— В устройствах NSG–700, NSG–900 без расширенной энергонезависимой памяти (модуля DoC или FLEX) вся файловая система хранися во Flash ROM в сжатом виде. При старте системы она распаковывается на виртуальный диск, создаваемый в оперативной памяти. По этой причине все изменения, сделанные в ходе работы устройства, утрачиваются при перезагрузке.
Сохранены могут быть только изменения, сделанные в директории /etc. По этой причине все пользовательские настройки, например, файл меню SMS-управления, также должны располагаться только в данной директории.

— В устройствах NSG–700, NSG–800, NSG–900 с модулем Doc или FLEX, а также NSG–1000, файловая система, за исключением /tmp и /var, постоянно хранится в расширенной энергонезависимой памяти в развёрнутом виде. Директории /tmp и /var создаются на виртуальном диске при старте системы и при перезагрузке полностью утрачиваются. Пользователь может создавать собственные директории, например, для хранения своих собственных программ (см. п.6.6); рекомендуется использовать для этой цели директорию /root.

§6.2.3. Сохранение изменений

Для сохранения изменений в конфигурации устройства, а также других изменений в файловой системе, следует использовать команду

```
savecfg
```

в командной оболочке Linux, или команду

```
write file
```

в основной командной оболочке. Обе команды выполняют одни и те же действия:

— Генерируют из текущей конфигурации, находящейся в proc-файловой системе, файл /etc/Zebra.conf .
— Архивируют и сжимают директорию /etc и записывают полученный архив во Flash ROM (только на устройствах NSG–700, NSG–900 без расширенной энергонезависимой памяти).

§6.2.4. Резервирование и восстановление конфигурации

Конфигурация устройства представляет собой набор текстовых файлов, хранящихся в директории /etc. Для резервирования следует сохранить ее на удаленном хосте при помощи FTP, TFTP или NFS. Примеры:

```
cd /tmp
tar –czf nsgconfig.tar.gz /etc
tftp 192.168.0.2 –p –l nsgconfig.tar.gz –r nsgconfig.tar.gz

mount –t nfs –o nolock 192.168.0.2:/config_backups /mnt/nfs
cd /tmp
tar –czf /mnt/nfs/ nsgconfig.tar.gz /etc
```

Для восстановления конфигурации из резервной копии следует загрузить и распаковать соответствующий файл любым из перечисленных способов. Примеры:

```
cd /tmp
tftp 192.168.0.2 –g –l nsgconfig.tar.gz –r nsgconfig.tar.gz
cd /
tar –xzf /tmp/nsgconfig.tar.gz

mount –t nfs –o nolock 192.168.0.2:/config_backups /mnt/nfs
cd /
tar –xzf /mnt/nfs/ nsgconfig.tar.gz
```

ПРИМЕЧАНИЕ Для упорядочения архива конфигураций, файлы с резервными копиями рекомендуется именовать в соответствии с административными именами устройств, установленными командой hostname.

§6.3. Сценарии Linux

§6.3.1. Стартовые скрипты Linux

Как и любая *nix-система, устройства NSG позволяют использовать развитый язык скриптов для командной оболочки. Например, нижеприведённый скрипт посылает *ping* на удалённый хост. В случае 4 неудачных попыток по 30 сек. подряд устройство рестартует.

```
#!/bin/sh
(
PORT=s1
RETRIES=4
PINGADDR=194.87.0.50
sleep 60
i=$RETRIES
while [ $i -gt 0 ]; do
   sleep 30
   if ip link show $PORT | grep -q -e '\<UP\>'; then
      :
   else
      i=$(($i-1))
      continue
   fi
   if ping -c 1 $PINGADDR; then
      i=$RETRIES
      continue
   else
      i=$(($i-1))
      continue
   fi
done >/dev/null 2>&1
sync
reboot
)&
```

Скрипты могут запускаться на исполнение либо вручную, либо при старте системы. Первое имеет смысл только в процессе отладки и настройки системы, поскольку устройства NSG по сути своей предназначены для непрерывной работы в необслуживаемом режиме. Что касается второго варианта, то команды автозапуска, по умолчанию, содержатся в файле /etc/rc.sh. При необходимости данный файл можно просмотреть или изменить средствами ОС Linux, например, с помощью редактора nano.

Кроме того, пользователь может создавать скрипты с другими стандартными именами и в стандартных директориях, предусмотренных для этой цели, например:

```
cat >/etc/init.d/S99mon
   вставить скрипт
   Enter
   Ctrl+D
chmod +x /etc/init.d/S99mon
savecfg
```

Такой скрипт автоматически начнёт выполняться (последним из скриптов S*NN*mon) при старте системы.

ВНИМАНИЕ При составлении скриптов следует учитывать, что стандартные стартовые скрипты Linux выполняются *до* загрузки конфигурации, заданной средствами основной командной оболочки. С другой стороны, скрипты, определённые в основной командной оболочке, исполняются *после* загрузки всей остальной конфигурации.

§6.3.2. Исполнение команд NSG в сценариях Linux

При исполнениии сценариев Linux в них могут быть включены не только команды собственно ОС Linux, но и команды, входящие в командную оболочку NSG. Для исполнения последних следует использовать утилиту config-nsg, параметром которой является текст команды, например:

```
config-nsg port eth0 ip ad 10.0.0.1/8
config-nsg card s1 im-v35
```

Как видно из первого примера, ключевые слова и параметры команд можно сокращать до тех пор, пока они являются однозначными.

В одном сценарии может использоваться несколько последовательных команд config-nsg, например:

```
config-nsg port s1 adm-state up; sleep 1; config-nsg led l1 client test-on
```

§6.3.3. Исполнение команд NSG через proc-файловую систему

Текущая конфигурация устройства, представленная командами основной оболочки, физичсеки существует в виде директорий и файлов в директории /proc/sys/nsg. При необходимости они могут быть считаны и даже перезаписаны командами и скриптами Linux, например:

cat /proc/sys/nsg/port/eth0/ip/address
: Просмотреть IP-адрес порта eth0

echo out3 > /proc/sys/nsg/port/s1/short
: Замкнуть контактную пару 3 модуля IM–DIO–2, установленного в разъём расширения s1

echo test-sos > /proc/sys/nsg/led/l1/client
: Включить светодиодный индикатор l1 в режиме SOS (3 коротких вспышки — 3 длинных — 3 коротких — пауза).

§6.3.4. Скрипты Linux в командной оболочке NSG

Основная командная оболочка NSG имеет встроенные средства для создания и исполнения скриптов Linux. Для этой цели предназначена вспомогательная команда script, доступная из меню (config-nsg)#. Команда используется следующим образом:

script add *номер* "*команда*"
: Создать/изменить запись с указанным номером в таблице сценариев. Номер записи может находиться в пределах от 1 до 1000. Текст команды может иметь длину до 255 символов и содержать любую команду ОС Linux. В частности, допускается определять переменные, используемые в последующих командах, и т.п.
Поскольку команда обычно содержит пробелы, она заключается в кавычки.

script exec { *номер* | –1 | all }
: Выполнить одну команду с указанным номером; если значение параметра равно –1 или all — выполнить последовательно все команды в порядке их нумерации.

script del *номер*
: Удалить команду с указанным номером из таблицы сценариев.

Сценарии, созданные командой script, последовательно отрабатываются при старте системы, а также могут быть выполнены индивидуально при наступлении ряда других событий (см. последующие параграфы данного раздела). С помощью команды script можно, в частности, стартовать дополнительные службы, не запускаемые по умолчанию, либо, наоборот, остановить службы, запущенные по умолчанию, но не требуемые в данном случае.

При составлении скриптов необходимо учитывать следующие особенности их вызова из командной оболочки NSG:

1. Команда script exec выполняется *после* загрузки всей остальной конфигурации. Таким образом, если скрипт должен запускать некоторую службу, а в конфигурации имеются команды, относящиеся к этой службе, то при загрузке устройства они будут проигнорированы (поскольку на этот момент служба еще не стартовала), а затем служба будет запущена с настройками, принятыми по умолчанию. Для корректной загрузки таких служб следует включать их в стартовые файлы собственно Linux (см. п.6.3.1).
2. Если сценарий предназначен *только* для исполнения по событию и не должен исполняться при старте системы, то необходимо внутри самого сценария организовать соответствующую проверку, например, с помощью каких-либо переменных окружения, специфических для вызова данного скрипта.

3. Команда script exec запускает указанный сценарий (или последовательность сценариев) на исполнение и ждет завершения его работы. По этой причине нельзя включать в сценарий бесконечные циклы, команды, ожидающие ввода с терминала и т.п. — они приведут к остановке работы командной оболочки.
4. Команда script exec ожидает сигнала о завершении исполнения команды. Если исполняемая команда запускает некоторый демон Linux, то в сценарий необходимо включить команду, выполняющую какой-нибудь фиктивный вывод, например:

   script add 1 "echo OK; syslogd"

5. При запуске демонов (или других программ, работающих в фоновом режиме) необходимо перенаправить их вывод в некоторый файл или в null. Вывод в стандартное устройство (stdout) для таких процессов невозможен.

§6.3.5. Исполнение сценариев при изменении состояния интерфейса

Для исполнения заданных скриптов при изменении состояния IP-интерфейсов следует использовать следующую команду в меню портов, VLAN, DLCI, туннелей и мостов:

state-script <номер>
: Указатель на номер сценария, вызываемого при возникновении какого-либо события на интерфейсе.

При вызове сценария ему передаются две переменные окружения: $NSG_IFACE_NAME — имя интерфейса (например, s1.0, s2.234, tuni1, eth0 и т.п.) и $NSG_IFACE_EVENT — целое (*integer*) значение из списка :

1	UP — переход интерфейса в состояние UP
2	DOWN — переход интерфейса в состояние DOWN
3	REBOOT — начало перезагрузки системы
4	CHANGE — изменение любого параметра IP-интерфейса (адреса и т.п.)
5	REGISTER — создание интерфейса в системе
6	UNREGISTER — удаление интерфейса из системы
7	MTU — изменение MTU
8	ADDR — изменение/удаление/добавление адреса
9	GOINGDOWN — состояние непосредственно перед переходом в DOWN

Примеры сценариев.

Следующий сценарий ведет журнал состояния интерфейсов:

```
script add 5   "echo $NSG_IFACE_NAME   $NSG_IFACE_EVENT >> /tmp/port_log"
```

Следующий сценарий ждет события UP на интерфейсе и добавляет маршрут в таблицу маршрутизации:

```
script add 6 "if [ $NSG_IFACE_EVENT = 1 ]; then ip rule add dev s1.0 table 10; ip route add default dev eth0 table 10; fi;"
```

ВНИМАНИЕ Сценарии выполняются асинхронно по отношению к состоянию интерфейса, но в строгой последовательности. Т.е. возможна ситуация, когда порт уже снова перешел в состояние DOWN, а вызывается еще только сценарий UP, за которым будет вызван сценарий DOWN. В случае "дрожания" состояния порта буфер хранит не более 10 событий и обеспечивает свертку повторяющихся событий. Например, последовательность событий UP–DOWN–UP–DOWN превратится в UP–DOWN.

Пример более сложного сценария и его применения. Если GPRS-соединение не удаётся установить в течение 2 мин (4×30 сек) после обыва или первой неудачной попытки, то устройство рестартует.

```
port s1
    encapsulation ppp
    virtual-template 1
    chat-script GPRS
    state-script 5
    exit
script add 5 "if [ x$NSG_IFACE_EVENT = x6 ]; then i=4; while [ $i -gt 0 ]; do sleep 30; if { ip link show
$NSG_IFACE_NAME | grep -q -e '[<,]\\+UP[,>]\\+'; }; then break; fi; i=$(($i-1)); continue; done; if [ $i -le 0 ];
then reboot; fi; fi >/dev/null 2>&1"
```

§6.3.6. Исполнение сценариев по непрохождению *ping*

Функция netping позволяет организовать регулярную посылку *ping* на заданный IP-адрес. Для более надёжного обнаружения отказа запросы можно посылать сериями, по несколько запросов в одной попытке. Как только на один их запросов получен ответ, попытка считается успешной и дальнейшие запросы не посылаются, для экономии трафика. Если ответ не получен ни на один запрос, то попытка считается неудачной. После заданного числа неудачных попыток подряд хост считается недоступным, и в этом случае выполняется заданный failure-script. После этого попытки послать *ping* продолжаются по прежнему графику; после первой удачной попытки считается, что соединение восстановлено, и выполняется соответствующий restore-script.

Внутри сценария, обрабатывающего события, могут быть использованы следующие переменные окружения:

NET_PING_EVENT со значением FAILURE либо RESTORE
NET_PING_DESTINATION_IP
NET_PING_SOURCE_IP

Подробно о настройке netping см. Часть 4.

§6.3.7. Исполнение сценариев по заданному расписанию

Функциональность в разработке. Предполагается, что в ближайших версиях NSG Linux будет реализована возможность исполнять скрипты в установленное время суток, или с установленной периодичностью. В настоящее время это возможно делать стандартными средствами Linux — с помощью crond.

§6.3.8. Исполнение сценариев по срабатыванию датчиков

Функциональность в разработке. Предполагается, что в ближайших версиях NSG Linux будет реализована возможность контролировать:

— Срабатывание дискретных датчиков, подключённых ко входам модулей IM–DIO–2, IM–1W или к асинхронным портам через внешние адаптеры RS–232/1–Wire.

— Нажатие программируемых кнопок на отдельных моделях и модификациях устройств NSG.

— Преодоление заданных пороговых значений на входах аналого-цифровых преобразователей (АЦП), подключённых к шине 1–Wire.

— Преодоление заданных пороговых значений температуры на датчиках, подключённых к шине 1–Wire.

— Преодоление контролируемых параметров на других специализированных датчиках.

— Отсутствие ожидаемых событий.

При выполнении этих условий будет выполняться event-script, заданный в настройках данного датчика.

§6.4. Стандартные команды и утилиты Linux

Данный раздел содержит указания относительно некоторых стандартных компонент Linux, знакомство с которыми (хотя бы поверхностное) может быть полезно администратору устройства NSG.

§6.4.1. dmesg

Команда dmesg выводит диагностические сообщения системы. Может быть полезной для анализа диагностики, выводимой при старте системы, или для отладки работы USB-устройств и модулей. Пример вывода при подключении USB-модема, известного системе:

```
.....
hub.c: new USB device <NULL>-1.1, assigned address 5
ttyACM1: USB ACM device
```

Пример вывода для не поддерживаемого USB-устройства:

```
.....
hub.c: new USB device <NULL>-1.1, assigned address 5
usb.c: USB device 5 (vend/prod 0xb05/0x1706) is not claimed by any active driver.
```

§6.4.2. syslog

Для мониторинга работы системы и отладки проблемных ситуаций может потребоваться анализ системного журнала. Служба Syslog включается в командной оболочке Linux, в простейшем случае, командой

```
syslogd
```

По умолчанию, весь вывод команды направляется в файл /var/log/messages . Относительно других вариантов использования данной службы см. соответствующие *man pages*.

Пример сценария, загружающего syslogd при старте системы:

```
script add 1 "echo OK; syslogd"
```

ПРИМЕЧАНИЕ Для наиболее частой практической задачи, требующей анализа Syslog — отладки соединений PPP, PPPoE, PPTP — функциональность syslogd в основном дублируется непосредственно в командной оболочке NSG. Команды chat-log и ppp-log доступны в меню соответствующего порта или туннеля (см. Часть 3, Часть 5).

§6.4.3. ping и traceroute

Стандартные команды для анализа функционирования IP-сети с помощью пакетов ICMP Echo Request/Reply. По сравнению с одноимёнными командами в основной командной оболочке NSG, имеют несколько более широкие возможности и набор параметров. В частности, в данной версии NSG Linux только они позволяют устанавливать произвольный адрес источника в отсылаемых пакетах.

§6.4.4. route

Команда route используется для настройки IP-маршрутизации непосредственно в ядре Linux. Рекомендуется использовать данную команду без параметров, только для просмотра текущей таблицы маршрутизации. Настройку маршрутизации следует производить средствами основной командной оболочки NSG.

§6.4.5. pppd

Демон для установления соединений PPP и производных от него протоколов — PPPoE, PPTP. Де-факто запускается средствами основной командной оболочки; ручная настройка средствами Linux не требуется и противопоказана, из соображений целостности системы. Тем не менее, любопытствующему пользователю рекомендуется просмотреть *man pages* и ознакомиться с обширным списком опций, доступных для данного демона. Если некоторая опция не настраивается штатными командами основной оболочки, то в случае необходимости большинство из них может быть настроено с помощью команды options в virtual-template, например:

```
ppp options "nobsdcomp local"
```

Исключение могут составлять отдельные "тяжеловесные" опции, например, multilink, которые могут быть исключены из текущей штатной версии NSG Linux по усмотрению разработчиков.

§6.4.6. arp

Команда arp предназначена для просмотра таблицы ARP, настройки статического ARP и ARP proxy. Рекомендуется использовать её только для первых двух задач. ARP proxy настраивается средствами основной командной оболочки NSG.

§6.4.7. iptables

Пакет IPtables предлагает широкие возможности для манипулирования IP-пакетами, включая разнообразные варианты NAT, коммутации и т.п., выходящие за рамки типовых общеупотребительных настроек.

Пример настройки Destination NAT средствами IPtables и скриптов (исключительно учебный; реальную конфигурацию удобнее настраивать штатными средствами основной командной оболочки, см. Часть 4). Физический порт s1 подключен к внешней сети, порт eth0 — к внутренней сети с FTP-сервером. Требуется обеспечить доступ к этому серверу из внешней сети в обоих режимах FTP (активном и пассивном).

```
!
nsg
    port s1
        ip address 123.145.167.189/30
        nat source masquerade
        exit
    port eth0
        ip address 10.0.0.1/8 anycast 0.0.0.0
        exit
    script add 1 "iptables -t nat -A PREROUTING -p TCP -i s1 --dport 20 -j DNAT --to-destination 10.0.0.2:20"
    script add 2 "iptables -t nat -A PREROUTING -p TCP -i s1 --dport 21 -j DNAT --to-destination 10.0.0.2:21"
    script exec -1
```

ПРИМЕЧАНИЕ Для настройки любых механизмов NAT при помощи скриптов необходимо, чтобы предварительно были загружены соответствующие модули ядра Linux. В данном примере, и в большинстве практических случаев, это делается в ходе исполнения команды

```
nat source masquerade
```

Описание команд IPtables см. в *man pages* по данному пакету.

§6.4.8. telnet и ssh

Стандартные команды для удалённого управления. Могут использоваться, например, для последовательного доступа "по цепочке" на устройство, если нарушена маршрутизация и доступны только хосты, находящиеся в непосредственно подключённых сетях.

Как частный случай, могут использоваться для обращения к самому устройству NSG, например, к асинхронному порту Reverse Telnet:

```
telnet 127.0.0.1 10023
```

Такое подключение может быть удобно, например, для первичной настройки и отладки сотовых модемных модулей.

§6.4.9. telnetd и sshd

Стандартные сервера указанных служб. Функциональность telnetd практически полностью контролируется настройками в основной командной оболочке (как для удалённого управления устройством, так и для доступа к физическим портам в режиме Reverse Telnet), поэтому его настройка требуется только в редких случаях.

С помощью сервера SSH можно организовать безопасный доступ к асинхронным портам устройства. (В данной версии — только средствами Linux.) Такую функциональность можно назвать Reverse SSH, по аналогии с Reverse Telnet.

§6.5. Доработанные и фирменные утилиты NSG Linux

§6.5.1. Трассировка трафика — tcpdump

Для трассировки трафика и отладки проблемных соединений в состав программного обеспечения входит утилита tcpdump. В NSG Linux с её помощью возможно производить трассировку не только пакетов IP, но также и пакетов X.25 и XOT.

ПРИМЕЧАНИЕ Утилита tcpdump не поставляется в дистрибутивах nsg700-linux-sumo8.bin и nsg900-linux-sumo.bin.

tcpdump запускается из командной оболочки Linux следующей командой:

tcpdump [–v] –i { *ip-интерфейс* | *X.25-порт* | xot }

где обязательным параметром является имя трассируемого объекта (sN, eth0, eth0.N, xot и т.п.).

Опция –v для порта X.25 означает, что уровень LAPB настроен как DCE; отсутствие этой опции означает, что уровень LAPB настроен как DTE. Данный параметр критически важен для корректного декодирования поля адреса LAPB (com/rsp).

При необходимости могут использоваться дополнительные опции, в частности:

–x или –X Вывод пакетов в шестнадцатиричном виде.

–s *число* Количество выводимых байт. При этом указанное число байт должно быть на 16 больше желаемого количества. Например, для вывода 133 байт следует вводить:
tcpdump –s 149 –x –i sN.

–w *файл* Сохранить трассу в файле в двоичном виде. Полученная трасса может быть импортирована в программы анализа сетевого трафика, такие как Ethereal/Wireshark. Данную опцию следует использовать в проблемных ситуациях, чтобы снять трассу и отправить её в службу технической поддержки NSG для изучения.

Остальные опции можно посмотреть командой

tcpdump --help

или в *man pages* по tcpdump.

Для порта X.25 записи выводятся в следующем формате:

direction com_rsp frame_type Pn s/r – lcn packet_type S/R MQD data_hex (leng)

где:

direction Направление передачи (In | Out)

Декодирование фрейма LAPB

com_rsp Команда или ответ (com | rsp)
frame_type Тип фрейма LAPB (SABM, UA, DISC, DM, INFO, RR, RNR, REJ, FRMR)
Pn Значение P-бита (P0 | P1)
s Значение поля N(S)
r Значение поля N(R)

Декодирование пакета X.25

lcn Номер логического канала
packet_type Тип пакета (CALL_REQ, CALL_ACC, DATA CLEAR_REQ, CLEAR_CONF, RR, RNR, ...)
S Значение поля P(S)
R Значение поля P(R)
MQD Значение битов M, Q, D. Обозначается наличием или отсутствием соответствующей буквы
data_hex Значение поля данных в шестнадцетиричном виде (первые 6 байт)
leng Длина поля данных в байтах

Пример вывода:

```
20:44:13.940394    In   com  SABM          P1
20:44:13.940608    Out  rsp  UA       P1
20:44:13.941493    Out  rsp  InFO     P0 0/0 –   0 RESTART_REQ    00 00 (2)
20:44:13.945399    In   com  InFO     P0 0/0 –   0 RESTART_REQ    00 00 (2)
20:44:15.942255    In   rsp  RR       P0 -/1
20:44:15.942622    Out  rsp  RR       P0 -/1
20:44:16.594610    Out  rsp  InFO     P0 1/1 –   1 CALL_REQ       02 77 06 42 07 07 (13)
20:44:16.603521    In   com  InFO     P0 1/2 –   1 CALL_ACC       00 06 42 07 07 43 (8)
20:44:16.616520    In   com  InFO     P0 2/2 –   1 DATA  0/0      0D 0A (2)
20:44:16.618534    In   com  InFO     P0 3/2 –   1 DATA  1/0      0D 0A (2)
20:44:16.618809    Out  rsp  InFO     P0 2/4 –   1 RR      -/2
20:44:16.629489    In   com  InFO     P0 4/3 –   1 DATA  2/0      4C 6F 67 69 6E 20 (45)
20:44:16.631503    In   com  InFO     P0 5/3 –   1 DATA  3/0      0D 0A (2)
20:44:16.631747    Out  rsp  InFO     P0 3/6 –   1 RR      -/4
20:44:16.642489    In   com  InFO     P0 6/4 –   1 DATA  4/0  Q   02 01 00 02 00 03 (43)
20:44:16.680450    In   com  InFO     P0 7/4 –   1 DATA  5/0      6E 73 67 20 6C 6F (11)
20:44:16.680694    Out  rsp  InFO     P0 4/0 –   1 RR      -/6
20:44:27.446777    In   com  InFO     P0 4/0 –   1 DATA  2/1      0D 0A (2)
20:44:27.476743    In   com  InFO     P0 5/0 –   1 DATA  3/1 M    68 6F 73 74 6E 61 (128)
20:44:27.476987    Out  rsp  InFO     P0 0/6 –   1 RR      -/4
20:44:27.498714    In   com  InFO     P0 6/1 –   1 DATA  4/1 M    70 72 6F 74 6F 22 (128)
20:44:27.516687    In   com  InFO     P0 7/1 –   1 DATA  5/1 M    74 65 20 61 64 64 (128)
20:44:27.516931    Out  rsp  InFO     P0 1/0 –   1 RR      -/6
20:44:27.539665    In   com  InFO     P0 0/2 –   1 DATA  6/1 M    2E 31 37 2F 38 20 (128)
20:44:27.556632    In   com  InFO     P0 1/2 –   1 DATA  7/1 M    61 63 65 20 65 74 (128)
20:44:27.556906    Out  rsp  InFO     P0 2/2 –   1 RR      -/0
20:44:27.565634    In   com  InFO     P0 2/3 –   1 DATA  0/1      6E 74 0D 0A 21 0D (26)
20:44:27.568624    In   com  InFO     P0 3/3 –   1 DATA  1/1      72 6F 6F 74 40 6E (16)
20:44:27.568899    Out  rsp  InFO     P0 3/4 –   1 RR      -/2
20:44:29.917077    In   com  InFO     P0 0/1 –   1 DATA  6/6      0D 0A (2)
20:44:29.923089    In   com  InFO     P0 1/1 –   1 CLEAR_REQ      00 00 (2)
20:44:29.923303    Out  rsp  InFO     P0 1/2 –   1 CLEAR_CONF
20:44:31.921898    In   rsp  RR       P0 -/2
20:44:33.918725    Out  rsp  RR       P1 -/2
20:44:33.923729    In   rsp  RR       P1 -/2
20:44:36.159306    In   com  DISC     P1
20:44:36.159520    Out  rsp  UA       P1
20:44:39.153462    Out  rsp  DM       P0
20:44:42.156162    Out  rsp  DM       P0
20:44:45.158862    Out  rsp  DM       P0
```

По умолчанию, трасса выводится в ту же консольную или телнет-сессию, в которой запущена tcpdump. При необходимости вывод можно перенаправить в файл стандартными средствами Linux:

```
tcpdump –i  s1 > /tmp/my.log.txt
```

и затем скопировать этот файл на ПК или просмотреть его:

```
cat /tmp/my.log.txt
```

§6.5.2. Сторожевой процесс — nsg-run-process

Утилита nsg-run-process всегда находится в списке исполняемых процессов, и её присутствие не должно удивлять пользователя. Это сторожевой процесс, задача которого — контролировать работу других процессов, и перезапускать их при необходимости. Например, при завершении сеанса PPP процесс pppd штатно завершается; именно nsg-run-process немедленно запускает его снова, с новым PID, и таким образом делает порт готовым к новому соединению.

Утилита запускается в ходе загрузки конфигурации в основную командную оболочку и не предназначена для употребления непосредственно пользователем.

§6.5.3. Программа эмуляции терминала — nsgcu

Утилита nsgcu (NSG Console Utility) — простая и компактная программа эмуляции терминала, предназначенная, в первую очередь, для использования в конвейерах и скриптах. С её помощью организуется, в частности, порт Reverse Telnet.

Формат команды для запуска программы:

nsgcu [опции] *порт*

Опции и параметры команды:

порт Имя порта (например, /dev|/nsg/a1).

–s *скорость* Скорость асинхронного порта, в бит/с; значение по умолчанию — 9600.

–t *формат* Формат асинхронной посылки, в виде {5|6|7|8}{N|O|E}{1|2}; значение по умолчанию — 8N1.

–E *символ* Спецсимвол для посылки ESC-кодов в терминальном режиме; символ по умолчанию — тильда (~).

--break *символ*
Специсимвол для посылки сигнала BREAK; по умолчанию не установлен.

--exit *символ*
Специсимвол для выхода из программы в терминальном режиме; по умолчанию не установлен.

–h, --help Вывод встроенной справки.

Символы --break и --exit используются сами по себе. После символа, определённого как –E, может быть введён один из следующих символов для непосредственного управления физическим интерфейсом:

. Завершить Telnet-соединение.
, Опустить сигнал DTR на 2 сек., чтобы принудить подключённый модем разорвать соединение.
Послать сигнал BREAK.
D Поднять сигнал DTR.
d Опустить сигнал DTR.
R Поднять сигнал RTS.
r Опустить сигнал RTS.
? Вывести список возможных escape-последовательностей на экран.

§6.5.4. Trivial PAD — tpad

Утилита tpad (Trivial PAD) — сборщик-разборщик пакетов, предназначенный для преобразования неструктурированного потока данных в пакеты X.25. С помощью данной программы организуется порт PAD.

tpad представляет собой простейшую реализацию PAD с ограниченным набором возможностей. Доработка программы предполагается по мере потребности.

Формат команды для запуска программы:

tpad [опции]

Опции команды:

–r *удалённый_X.121_адрес*
X.121 адрес, который будет использоваться для автоматического установления соединения по поднятию сигнала DCD (в текущей версии не реализовано) или при инициализации порта, если DCD в этот момент уже поднят. Данный адрес будет подставляться в пакеты CALL, отправляемые с данного порта, в качестве вызываемого адреса (Called Address). Адрес может содержать до 15 десятичных цифр.
Если адрес не установлен (в качестве значения при этом выводится $), то никакое соединение автоматически не устанавливается.

–l *локальный_X.121_адрес*
X.121 адрес, который будет подставляться в пакеты CALL, отправляемые с данного порта, в качестве вызывающего адреса (Calling Address). Адрес может содержать до 15 десятичных цифр. Если адрес не установлен (в качестве значения при этом выводится $), то вызывающий адрес в пакетах CALL отсутствует.

–d *порт*
Устройство ввода-вывода, с которым будет работать утилита. При подключении к данному устройству пользователь получит приглашение PAD, после чего он может ввести адрес X.121 и установить коммутируемое логическое соединение с удалённым узлом сети. (При условии, что в

устройстве определён маршрут к вызываемому узлу, см. Часть 3.)
Если опция –d не указана, то по умолчанию tpad работает со стандартными потоками stdin, stdout, т.е. при запуске программы без указания устройства пользователь получит приглашение (звёздочку) на свой терминал и далее будет работать в режиме PAD. Для завершения работы следует в командном режиме PAD набрать команду quit.

–t *секунды*

Максимальное время неактивности, по истечении которого соединение X.25 разрывается. Данный параметр является единым для командного режима и режима данных, на приём и на передачу. Значение по умолчанию — 300 сек.

–p { 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 }

Максимальная длина собираемого пакета (в байтах). Это один из критериев, по которым может производиться отправка пакета. Значение по умолчанию — 128.

–w *1 ... 7*

Размер окна пакетного уровня для соединений, устанавливаемых с данного порта. Передаётся в поле Facilities пакета CALL. Значение по умолчанию — 2.

--dtr *0 ... 60*

Управление сигналом DTR интерфейса RS–232 при разрыве сетевого соединения: сигнал опускается на указанное число секунд (по умолчанию — 2 сек.), затем поднимается снова. Это позволяет разорвать физическое соединение, реинициализировать модем и т.п. Если установлено значение 0, то сигнал DTR поднят постоянно.

--prompt *строка*

Строка приглашения, которая будет выводиться подключённому пользователю при работе в командном режиме PAD. Значение по умолчанию — звёздочка (*).

--par*N значение*

Параметры профиля PAD. В данной версии tpad поддерживаются следующие параметры:

Параметры согласно ITU–T X.3		Значение для профилей	
Номер	**Назначение**	**прозрачный**	**построчный**
1	Сигнал ВНИМАНИЕ (Recall character)	0	1
2	ЭХО (Echo)	0	1
3	Сигнал отправки пакета (Forwarding characters)	0	2
4	Тайм-аут отправки пакета (Idle timer delay)	0	0
5	Подчиненное устройство (Ancillary Device Control)	0	0
12	Управление потоком (Flow Control)	0	1
13	Вставка символа перевода строки (Linefeed Insertion)	0	6

По умолчанию установлен прозрачный профиль.
Если удалённая сторона при установлении соединения присылает определённый профиль PAD (например, прозрачный), то этот профиль будет принят и установлен.

§6.5.5. Клиент TACACS+ — nsgtaclient

Утилита nsgtaclient осуществляет аутентификацию на заданном сервере(-ах) TACACS+. Формат команды для запуска программы:

nsgtaclient *опция значение* [*опция значение ...*]

Возможные опции:

–s, --server Параметры сервера TACACS+, в формате *name*:*key*:*port*:*timeout*, где
- *name* IP-адрес сервера или его имя (если включена служба DNS)
- *key* Ключ для шифрования пакетов TACACS+
- *port* Номер порта TCP сервера; значение по умолчанию — 49
- *timeout* Время ожидания ответа от сервера, в секундах; значение по умолчанию — 5 сек.

Данная опция может встречаться в командной строке до 8 раз.
Если параметры *name* или *key* содержат двоеточие (:), то его необходимо заменить на последовательность \\: .

–l, --login Имя аутентифицируемого пользователя.

–p, --password

Пароль пользователя.

–P, --port Имя порта, к которому подключён пользователь.

–a, --avpair Пары атрибутов TACACS+ (в формате *атрибут=значение*). Данная опция может встречаться в командной строке до 255 раз.

–r, --retry Число попыток обращения к каждому серверу.

–d, --debug Вывод отладочной информации.

–h, --help Вывод встроенной справки.

Программа последовательно посылает по одному запросу к каждому серверу, в порядке их перечисления в командной строке. Если список исчерпан, а ответ не получен, то посылка запросов начинается снова с первого сервера, и повторится, в общей сложности, retry раз к каждому серверу. После первого ответа (положительного или отрицательного) от какого-либо сервера посылка запросов прекращается.

В случае неудачной аутентификации программа возвращает код ошибки, отличный от 0. Если от сервера получен список avpair, то он выводится в stdout.

§6.5.6. Обработчик SMS — nsgsms

Утилита nsgsms выполняет обработку SMS-сообщений. На стороне клиента при этом используется сотовый телефон с Java-приложением MoNsTer (MObile NSg TERminal). В совокупности они позволяют исполнять заданный набор команд, хранящийся в виде меню в файле /etc/nsgsms.conf (подробно о формате данного файла см. в Части 3). Меню может содержать произвольные команды для управления как самим устройством NSG, так и подключённым к нему оборудованием. Формат команды для запуска программы:

nsgsms [*опция значение* [*опция значение ...*]] *порт*

Параметры и опции:

порт Имя разъема расширения, в который установлен сотовый модуль, в терминах Linux (например, /dev/nsg/s1).

–s *скорость* Скорость асинхронного порта, в бит/с; значение по умолчанию — 115200.

–t *формат* Формат асинхронной посылки, в виде {5|6|7|8}{N|O|E}{1|2}; значение по умолчанию — 8N1.

–p { ppp | term }
Тип выводного потока. При использовании данной опции nsgsms будет запущена как труба: она будет обрабатывать SMS, а все остальные данные транслировать между устройством *порт* и своим стандартным потоком ввода-вывода. Труба может быть организована двумя способами:
- term Стандартный поток ввода-вывода есть обычный терминал. На него будут выводиться данные, идущие с модема, с него же можно вводить AT-команды.
- ppp nsgsms подключено как pty-устройство к демону pppd. При этом модем будет работать в режиме передачи данных, и одновременно nsgsms будет его прослушивать на предмет прихода SMS и вставлять свои исходящие SMS в поток данных, направлемый в модем.

В отсутствие данной опции nsgsms использует порт в монопольном режиме и обрабатывает исключительно SMS-трафик.

–n *число* Максимальное число SMS, которые могут быть отправлены в ответ мобильному пользователю в случае обширного вывода (статистики портов и т.п.).

–u *телефон* Телефонный номер мобильного клиента, в том формате, в котором он определяется сотовой сетью. nsgsms обрабатывает и отвечает только на сообщения с заданных номеров, остальные игнорируются. Всего в данной реализации поддерживается до 32 пользователей (включая как заданных опцией -u, так и указанных в файле nsgsms.conf).
Если значение опции короче, чем строка, выводимая АОН, то SMS-управление доступно для любых клиентов, у которых начальная часть номера совпадает с заданным значением.

–l *файл* Вывод журнала работы в файл.

–v, –v, –vvv Три уровня детализации выводимых сообщений о работе программы.

–d Запуск в качестве резидентной программы (демона). Данная опция несовместима с -p.

--exit *символ* Выбор спецсимвола, который будет использоваться в терминальном режиме (-p term) для выхода из программы. Значение по умолчанию — пустое, вводится как два апострофа подряд ('').

--timestamp Включать в каждую запись журнала отметку времени.

–h, --help Вывод встроенной справки.

§6.6. Участие пользователей в развитии программы NSG Linux

Пользователи, обладающие необходимыми знаниями и навыками в области программирования для ОС Linux, могут самостоятельно реализовать специфические программные возможности, требуемые для решения их задач, а также внести свой вклад в развитие проекта в целом. С этой точки зрения, устройства NSG под управлением NSG Linux можно рассматривать как Linux-машину общего вида, на которой, наряду с коммуникационным программным обеспечением NSG, исполняются дополнительные приложения пользователя.

Для самостоятельной разработки приложений необходимо загрузить и установить Embedded Linux Development Kit (ELDK) комании Denx Software Engineering:

http://www.denx.de/

Рекомендуется использовать версию 3.1.1 во избежание расхождений версий файлов и библиотек. С помощью этого инструментария пользователи могут самостоятельно разрабатывать специфические приложения для своих нужд, а также переносить на эту платформу программные продукты, доступные в исходных кодах, при условии, что такое их использование не нарушает права интеллектуальной собственности третьих лиц. В частности, пользователи могут переносить на устройства NSG имеющиеся у них программные продукты собственной разработки, продукты, распространяемые с открытым кодом, и продукты, законно приобретенные пользователями у сторонних разработчиков с правом дальнейшего изменения.

Приложения пользователя рекомендуется устанавливать в файловую систему, расположенную в развернутом виде на устройствах расширения энергонезависимой памяти (DoC или FLEX, USB Flash или USB HDD, в зависимости от модели шасси). При этом следует иметь в виду, что память типа FLEX и USB Flash предназначена, в основном, для хранения приложений. Для часто перезаписываемых пользовательских данных, таких как статистика, журналы ввода-вывода и т.п., следует использовать модуль DoC, выдерживающий бóльшее число циклов стирания-записи, или жесткий диск.

В устройствах, не оснащенных расширенной энергонезависимой памятью, рабочая файловая система располагается на виртуальном диске в оперативной памяти. Приложения пользователя рекомендуется устанавливать в директорию /root, которая есть линк на /etc/root. После установки необходимо выполнить команду (в командной оболочке ОС Linux):

```
savecfg
```

По данной команде вся директория /etc, упаковывается в один архив и записывается в энергонезависимую память — при условии, что ее объём достаточен для хранения дополнительных компонент.

Вопрос об участии сторонних разработчиков в задачах, связанных с развитием ядра NSG Linux и протокольных компонент системы, решается на основе индивидуальных соглашений.