



Маршрутизаторы NSG

Программное обеспечение NSG Linux 2.0

Руководство пользователя

Часть 7

Основные команды и утилиты NSG Linux

Версия программного обеспечения 2.0 build 7

Обновлено 18.04.2017

АННОТАЦИЯ


Данный документ содержит руководство по настройке и применению маршрутизаторов NSG, оснащенных программным обеспечением NSG Linux 2.0. Документ имеет следующую структуру:

- Часть 1. Общесистемная конфигурация.
- Часть 2. Настройка физических интерфейсов, портов и сетевых интерфейсов. Обработка трафика Ethernet.
- Часть 3. Обработка IP-трафика.
- Часть 4. Приложения и службы IP.
- Часть 5. Туннелирование и виртуальные частные сети (VPN).
- Часть 6. Система обеспечения бесперебойных соединений *uiTCP*.
- Часть 7. Основные команды и утилиты NSG Linux.

Руководства по применению маршрутизаторов под управлением NSG Linux 1.0 и базового программного обеспечения NSG, а также других продуктов NSG (модемов, мостов и т.п.), содержатся в отдельных документах.

ВНИМАНИЕ

Данное Руководство пользователя предназначено для лучшего понимания процедуры настройки в целом и описывает суть выполняемых действий — а именно, *что* необходимо настраивать. Рассматриваемые вопросы относятся, как правило, к сути используемой технологии и являются общими для любых её реализаций, независимо от конкретного производителя и устройства. (Исключением являются вопросы, специфичные для оборудования NSG — таких, как организация пользовательского интерфейса или настройка системы бесперебойных соединений *uiTCP*.)

Основной документацией по NSG Linux 2.0 является встроенная справка на борту устройства. Она описывает конкретные команды и параметры настройки — т.е. *как* настраивать функции и возможности, описанные в данном Руководстве. Для просмотра справки по каждому из параметров следует использовать кнопку  в Web-интерфейсе или команду `_manual (_m)` в консольном интерфейсе. Если справка на вашем языке отсутствует, следует установить на устройстве русскую локаль.

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» 2009–2017

ООО «Эн-Эс-Джи»
Россия 105187 Москва
ул. Вольная, д.35
Тел./факс: (+7-495) 727-19-59 (многоканальный)

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

§ СОДЕРЖАНИЕ §

Часть 7. Основные команды и утилиты NSG Linux.

§7.1. Основы работы в ОС Linux	4
§7.1.1. Документация по ОС Linux	4
§7.1.2. Вход в систему и переключение между оболочками	4
§7.1.3. Командная строка	5
§7.1.4. Особенности командного языка *nix для пользователей других ОС	5
§7.1.5. Базовые файловые операции	6
§7.1.6. Стандартные потоки и перенаправление ввода-вывода.....	7
§7.1.7. Просмотр и фильтрация текстовых файлов	7
§7.1.8. Создание и редактирование текстовых файлов	8
§7.1.9. Приём и передача файлов	8
§7.1.10. Переменные окружения	8
§7.1.11. Системная дата и время	8
§7.1.12. Системные операции.....	9
§7.1.13. Сохранение изменений и перезагрузка устройства.....	9
§7.2. Особенности устройств NSG как Linux-систем	10
§7.2.1. Структура файловой системы NSG Linux	10
§7.2.2. Ручное резервирование и восстановление конфигурации	10
§7.3. Сценарии Linux	11
§7.3.1. Стартовые скрипты Linux.....	11
§7.3.2. Исполнение команд nsgsh в сценариях Linux	12
§7.3.3. Исполнение сценариев по результатам тестов (<i>ping</i> и др.).....	14
§7.3.4. Исполнение сценариев по событию и по заданному расписанию	15
§7.3.5. Особенности использования сценариев для рестарта.....	15
§7.3.6. Исполнение скриптов в качестве демонов	15
§7.4. Стандартные команды и утилиты Linux	17
§7.4.1. dmesg	17
§7.4.2. syslog.....	17
§7.4.3. uptime и top.....	17
§7.4.4. Утилиты для настройки IP.....	17
§7.4.5. iptables.....	17
§7.4.6. pppd.....	18
§7.4.7. ping и traceroute	18
§7.4.8. telnet и ssh.....	18
§7.4.9. Передача двоичных файлов по ssh.....	18
§7.4.10. Трассировка трафика — tcpdump.....	19
§7.5. Доработанные и фирменные утилиты NSG Linux	20
§7.5.1. Демон конфигурации — nsgconfd.....	20
§7.5.2. Утилита конфигурации — nsgsh	20
§7.5.3. Генератор тестового трафика — fox	20
§7.5.4. Программа эмуляции терминала — nsgcu.....	21
§7.5.5. Монитор состояния сигнала DCD — dcdstarter	22
§7.5.6. Обработчик SMS — nsgsms	23
§7.5.7. Передача SMS и исполнение AT-команд — at2	24
§7.5.8. Управление по шине 1-Wire — nsgow	25
§7.5.9. Отправка электронных писем — nsgsmtp	26
§7.5.10. Доступ к удалённому устройству и синхронизация конфигурации — nsgconfsync.....	26
§7.6. Участие пользователей в развитии NSG Linux	28

§7.1. Основы работы в ОС Linux

§7.1.1. 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.

§7.1.2. Вход в систему и переключение между оболочками

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 или SSH. После входа системное приглашение принимает вид:

```
hostname #
```

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

Из командной оболочки Linux всегда можно вызвать командную оболочку NSG при помощи команды `nsgsh`, а затем вернуться из `nsgsh` обратно с помощью команды `_quit`. Обратное невозможно, поскольку для пользователя `nsg` (и тем более для дополнительных пользователей) доступ с правами суперпользователя запрещён по соображениям безопасности и целостности системы.

Во многих случаях бывает удобно иметь одновременно два сеанса работы с системой — один как `nsg`, или через Web-интерфейс под любым из двух пользователей, другой в режиме командной строки как `root`. Например, подключиться к устройству через два сеанса Telnet, или через Telnet и Web. Все способы подключения к устройству допускают одновременную работу нескольких пользователей, в т.ч. под одними и теми же именами. При этом только один из них, имеющий административные права на чтение и запись, получает текущий статус `admin` и может изменять конфигурацию устройства с помощью `nsgsh` или Web-интерфейса. Все остальные пользователи имеют статус `user` и могут только просматривать конфигурацию. Однако на работу в `ash` это ограничение не распространяется, суперпользователь `root` в своей командной среде всегда имеет полный доступ к устройству.

§7.1.3. Командная строка

При вводе команд Linux можно перемещаться по вводимой командной строке с помощью клавиш ← и →, редактировать содержимое 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.

§7.1.4. 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:

<code>/</code>	Корневая директория файловой системы на данном устройстве
<code>./</code>	Текущая директория
<code>../</code>	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 некоторые другие спецсимволы, но не могут содержать символы `&` ; `(`) `>` `<` `|` . При необходимости спецсимволы могут указываться в виде *escape*-последовательностей, например, `\:` (двоеточие), но использовать эту возможность без крайней необходимости не рекомендуется.
5. Механизм ассоциаций между расширениями (суффиксами) файлов и приложениями, которыми следует открывать эти файлы, используется только в рамках интерактивных 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, по умолчанию, 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*.

§7.1.5. Базовые файловые операции

Для выполнения основных операций с файлами и директориями используются команды:

ls [-l] [путь] Вывести содержимое директории. В выводимом списке директории обозначаются жирным шрифтом, файлы — обычным. При указании опции *-l (long)* выводится расширенная информация, включая размеры файлов и их атрибуты.

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.

rmdir 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 к указанному файлу для всех пользователей. В Linux предусмотрены следующие 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 <code>gzip</code> ; в *nix-системах сжатие и архивирование традиционно являются двумя независимыми операциями.
j	При архивировании сжимать и восстанавливать файлы с помощью утилиты <code>bzip2</code> .
f архив	Имя архива (f ставится последней в пакете опций).

C директория Распаковывать корень архива в указанную директорию.

Первой из указанных опций должна стоять с 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*.

§7.1.6. Стандартные потоки и перенаправление ввода-вывода

Каждая программа при выполнении 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 команду 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`.

\$(команда)

`команда` Преобразовать вывод команды в текстовую строку, которая может быть присвоена переменной окружения, передана другой команде в виде параметра и т.п. (Во втором синтаксисе используются именно обратные, а не прямые, апострофы.)

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`.

§7.1.7. 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
```

Искомая подстрока может содержать пробелы, но в этом случае её необходимо взять в кавычки или апострофы.

Если указана опция `-v`, то выводятся строки, не содержащие указанную подстроку.

`tail` [-f] *файл*

Вывод последних строк файла (по умолчанию 10). Удобно для просмотра последних записей в длинных логах.

Если указана опция `-f`, команда продолжает выводить новые записи по мере их поступления.

Данный режим удобен для просмотра отладочных журналов в реальном времени. Для выхода следует нажать `CTRL-C`.

Подробно о синтаксисе и опциях данных команд см. встроенную справку (`-h`, `--help`) или соответствующие *man pages*.

§7.1.8. 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.7.1.9). 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
```

§7.1.9. 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 стандартные утилиты Linux:

<code>tftp</code>	Клиент TFTP
<code>ftpget</code>	Клиент FTP для приёма файлов
<code>ftpput</code>	Клиент FTP для передачи файлов
<code>wget</code>	Клиент HTTP

Чтобы получить 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.

§7.1.10. Переменные окружения

Переменные окружения широко используются в скриптах Linux. Значением переменной является текстовая строка. Переменные окружения могут получать значения при помощи оператора вида

```
ИМЯ = произвольная_текстовая_строка
```

где правая часть может быть как явно заданной строкой, так и результатом каких-либо вычислений (например, оператора `$(команда)`). Имя переменной может быть произвольным по усмотрению пользователя. Для удобства принято использовать в именах только заглавные буквы, цифры и подчёрк (`_`).

Значение переменной подставляется в другие команды следующим образом:

```
$ИМЯ
```

Если переменной не присвоено значение, то `$ИМЯ` преобразуется в пустую строку.

Просмотреть все переменные окружения, определённые в системе, и их значения можно командой `set`.

§7.1.11. Системная дата и время

Для просмотра и установки системного времени используется команда:

```
date [ММДДччммГГГГ[.cc]]
```


§7.1.12. 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 следующих команд:

`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.

`kill -9 PID`

`kill -15 PID` Завершить сложно зависшие процессы, если они не завершаются простой командой `kill`.

`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.

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

§7.1.13. Сохранение изменений и перезагрузка устройства

Для перезагрузки устройства средствами командной оболочки Linux следует использовать команду:

```
reboot
```

Изменения, сделанные пользователем в командной среде `ash`, сохраняются автоматически и не требуют особой команды. При программной перезагрузке они корректно копируются в энергоназависимую память.

Перед аппаратным рестартом (по питанию или кнопке `Reset`) устройства необходимо выполнить команду `sync`, чтобы гарантировать, что сохранённые изменения физически записаны в энергонезависимую память, а не остались в кэше.

§7.2. Особенности устройств NSG как Linux-систем

§7.2.1. Структура файловой системы NSG Linux

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

<code>/etc</code>	Содержит все настройки системы.
<code>/etc/private</code>	Специальная директория для файлов пользователя. Не изменяется при обновлении ПО в рабочем режиме (см. §7.3.1).
<code>/etc/private/init.d</code>	Директория для стартовых скриптов пользователя (см. §7.3.1).
<code>/root</code>	Предназначена для хранения файлов пользователя <code>root</code> .
<code>/var</code>	Содержит временные файлы и директории, создаваемые системой в ходе работы. Например, вывод системного журнала по умолчанию направляется в файл <code>/var/log/messages</code> .
<code>/tmp</code>	Предназначена для хранения временных файлов пользователя. Например, сюда можно поместить файл конфигурации системы в архивированном виде, передаваемый или принимаемый по <code>tftp</code> . Де-факто является ссылкой на <code>/var/tmp</code> .

Директории `/tmp` и `/var` при перезагрузке системы полностью утрачиваются.

Остальные физические директории не предназначены для работы пользователя.

<code>/proc</code>	Виртуальная файловая система, используемая для работы Linux. Некоторые файлы из неё могут потребоваться службе технической поддержки NSG для поиска неисправностей.
--------------------	---

§7.2.2. Ручное резервирование и восстановление конфигурации

Конфигурация устройства представляет собой набор файлов, хранящихся в директории `/etc`. Для резервирования можно вручную 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`.

Основная часть конфигурации, настраиваемая с помощью `nsgsh` или Web-интерфейса, хранится в файле `/etc/nsgconfig`. Это текстовый файл, однако он не предназначен для ручного редактирования пользователем; для работы с конфигурацией в текстовом виде следует использовать соответствующие инструменты в `nsgsh` или Web-интерфейсе. Отдельно от основной конфигурации хранятся пароли для доступа к устройству NSG (но не пароли PPP), ключи RSA и сертификаты X.509, а также скрипты и утилиты, созданные пользователем.

§7.3. Сценарии Linux

§7.3.1. Стартовые скрипты Linux

Как и любая *nix-система, устройства NSG позволяют использовать развитый язык скриптов для командной оболочки. Скрипты могут запускаться на исполнение вручную, при старте системы, или по исполнению некоторых условий, описанных в основной командной оболочке NSG (срабатыванию таймера, датчиков, [не]прохождению *ping*, и т.п.). Первое имеет смысл только в процессе отладки и настройки системы, поскольку устройства NSG по сути своей предназначены для непрерывной работы в необслуживаемом режиме.

Пользовательские скрипты следует хранить в директории `/etc/private/`. Она предназначена именно для хранения файлов, не контролируемых разработчиками NSG, и гарантированно сохраняется при обновлении ПО в рабочем режиме. 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 можно редактировать на ПК и переносить на устройство с помощью любых доступных средств (*ssh*, *fish*, *sftp*, *ftpget*, *tfpt*, *wget*), а также создавать прямо на борту устройства с помощью редактора *nano* или консольного ввода, например:

```
cat >/etc/private/my-monitor
    вставить скрипт
    Enter
    Ctrl+D
chmod +x /etc/private/my-monitor
savecfg
```

Скрипты, которые должны исполняться при старте системы, следует помещать в директорию `/etc/private/init.d` с расширением `.start`. Эти файлы будут выполняться в порядке лексикографического возрастания их имён (без учёта расширения `".start"`). Другие файлы в данной директории при старте игнорируются. Пример имён файлов:

```
2.start      — будет выполнен первым
24foo.start  — будет выполнен вторым
2fee.start   — будет выполнен третьим
skip.start~  — игнорируется, т.к. расширение отлично от '.start'
```

Пример. Устройство NSG-1000/GW с опциональным HDD (устройство `/dev/sdb`). На HDD размечен вручную раздел `/dev/sdb2` с файловой системой `ext2` для хранения журнала *uiTCP* (`/dev/sdb1` — основное ПО NSG Linux 2.0). При старте требуется смонтировать его на чтение и запись. Делать это посредством `/etc/fstab` настоятельно не рекомендуется, поскольку нельзя гарантировать, что при каком-либо очередном обновлении

ПО сохранится и сам этот файл, и каталог, указанный для монтирования; в этом случае их придётся создавать заново вручную. Поэтому лучше создать дополнительный скрипт инициализации системы. Назовём его `mnt-uitcp-logs.start`. Создаём файл и редактируем его с помощью `nano`:

```
nano /etc/private/init.d/mnt-uitcp-logs.start
```

Вводим в `nano` текст скрипта:

```
#!/bin/sh
mkdir -p /mnt/uitcp-logs || exit 1
mount -t ext2 -o rw,noatime /dev/sdb2 /mnt/uitcp-logs
```

Сохраняем файл, выходим из `nano`, устанавливаем файлу флаг исполняемого:

```
chmod +x /etc/private/init.d/mnt-uitcp-logs.start
```

Полученный скрипт `/etc/private/init.d/mnt-uitcp-logs.start` будет выполняться при каждом старте системы, в том числе и после удалённого обновления ПО. После этого в конфигурации `uiTCP` можно указать

```
uitcp
: configure
: : log
: : : logFile = "/mnt/uitcp-logs/uitcp"
```

ВНИМАНИЕ При составлении скриптов следует учитывать последовательность исполнения стартовых скриптов и конфигурации:

- 1) системные стартовые скрипты из директории `/etc/init.d/`
- 2) пользовательские стартовые скрипты из директории `/etc/private/init.d/`
- 3) конфигурация системы — файл `/etc/nsgconfig`

§7.3.2. Исполнение команд `nsgsh` в сценариях Linux

Командная оболочка `nsgsh` может исполняться из командной строки оболочки `ash` в пакетном режиме. Операндами при её вызове являются внутренние команды `nsgsh`, которые следует выполнить. После выполнения этих команд `nsgsh`, по умолчанию, завершает работу и возвращает управление `ash`. Формат вызова `nsgsh`:

```
nsgsh опции путь.параметр=значение [[путь.параметр=значение] ... ]
```

Возможные форматы операндов (переход в узел, установка параметра, от текущего узла, от корневого узла, от родительского узла и т.п.) описаны в [Части 1](#). В отличие от интерактивной работы в `nsgsh`, при пакетном вызове к перечню спецсимволов, требующих указания особым образом, добавляются спецсимволы `ash` (`$ > < | & `` и т.п.) — в путях, параметрах и значениях. Символ `/` в данном случае специальным не является.

ВНИМАНИЕ В отличие от версий NSG Linux 2.0 *build 4* и ранее, точка вводится так же, как и другие спецсимволы — т.е. в кавычках или в виде `esc`-последовательности `\`. Прежний формат (две точки подряд преобразуются в одну, вводимую буквально) более не работает.

Если вводятся сразу несколько операндов, они разделяются пробелами; в этом случае `nsgsh` обрабатывает последовательно каждый операнд как отдельную команду, независимо от результатов исполнения предыдущей.

ВНИМАНИЕ При написании скриптов необходимо учитывать алгоритмы работы `ash` и двойной разбор `esc`-последовательностей: сначала в `ash`, потом в самой `nsgsh`. Примеры:

```
# nsgsh .port.eth0.vlan.eth0\101._print
```

Последовательность преобразуется `ash` в точку, и в `nsgsh` передаётся

```
# nsgsh .port.eth0.vlan.eth0.101._print
```

Далее каждый узел в указанном пути разрешается в полное имя:

```
port → port
eth0 → eth0
vlan → vlan
eth0 → eth0.101 (если он единственный)
```

узел `101` оказывается лишним, и в результате выдаётся ошибка. Правильная нотация:

```
# nsgsh .port.eth0.vlan.eth0\\\101._print
```

Другой пример:

```
nsgsh .port.eth0.vlan.eth0\\\101\(_apply _print\)
```

В этом случае `nsgsh` получит `.port.eth0.vlan.eth0\\\101\(_apply и _print\)` как два последовательных операнда (и оба некорректные, с непарными скобками). Правильная нотация — необходимо закрыть пробел между ними:

```
nsgsh .port.eth0.vlan.eth0\\\101\(_apply\ _print\)
```

Изменения параметров автоматически не применяются ни в пакетном режиме, ни в интерактивном. Чтобы они вступили в силу, необходимо завершить командную строку командой `_apply` в нужном узле, например:

```
nsgsh .port.3g.main.attempts.1 .port.3g.aux.attempts.0 .port.3g._apply
```

Разовые команды исполняются немедленно и не требуют завершающего `_apply` .

Если команда требует подтверждения, то при исполнении её в пакетном режиме необходимо ввести это подтверждение в конце операнда, например:

```
nsgsh .system.reboot.yes
```

Это же относится и к некоторым другим случаям, когда команда требует какого-либо ввода — например, для рестарта сотового интерфейса с заданной SIM-картой:

```
nsgsh .port.edge.restart.main
```

Пример вызова:

```
nsgsh .port.eth0.link.adm.down .port.eth0._apply .po.eth1.li.ad.u .port.eth1._a
```

Опции команды:

- h, --help Вывести справку о команде.
- q Молчаливый режим, при этом выводятся только ответы исполняемых команд.
- i После выполнения команды не выходить в `ash`, а остаться в интерактивном режиме `nsgsh`. Вызов `nsgsh -i` без операндов равносильно просто вызову `nsgsh`.
- c Запустить оболочку в экранном режиме (имеет смысл только одновременно с `-i` либо без операндов).
- timeout= *секунды*
- t *секунды* Установить время ожидания ответа для команд, требующих длительного времени (*build* 6 и выше). В интерактивном режиме, если команда исполняется дольше установленного времени (по умолчанию — 5 сек), выводится предупреждение:

```
Print Q to quit nsg shell.
Operation too long. Waiting...
Operation too long. Waiting...
```

и пользователь сам решает: прервать операцию или ждать. В режиме скрипта такой возможности нет, поэтому операция прекращается безусловно. Увеличение таймаута позволяет гарантировать исполнение длительных операций полностью.
- sid= *SID* Запустить `nsgsh` с заданным идентификатором сессии. Параметр служебный, используется некоторыми внутренними скриптами, при этом нужный идентификатор определяется автоматически. Для ручного использования не имеет смысла, поскольку идентификатор нужной сессии априори неизвестен.

Права доступа при исполнении `nsgsh` в пакетном режиме. При исполнении `nsgsh` в пакетном режиме запускается отдельная сессия пользователя `root`, на которую распространяются общие правила открытия сессий для `nsgsh` и Web-управления. В частности, если в этот момент на устройстве уже имеется сессия со статусом `admin`, то сессия будет открыта с правами `read-only` и не сможет вносить изменения в конфигурацию.

Безусловно доступны для сессии со статусом `user`, вне зависимости от статуса текущей сессии:

- Встроенные команды просмотра конфигурации — `_show`, `_print` .
- Разовые команды просмотра журналов отдельных объектов, их статистики и состояния системы — `log`, `show`, `system.show.*` и др.
- Команды управления исполнительными компонентами устройства — встроенными светодиодами и внешними контроллерами, управляемыми по шине 1-Wire или USB.
- Разовые команды для рестарта отдельных объектов (например, `.port.m1.restart`), а также большинство других команд.
- Команды в узле `tools` .

К действиям, доступным только для сессии со статусом `admin`, относятся:

- Изменение значений любых параметров, кроме параметров исполнения разовых команд в узле `tools` .
- Создание и удаление узлов (элементов списков), установка значений параметров по умолчанию.
- Применение, сохранение, загрузка-выгрузка и синхронизация конфигурации.
- Перезагрузка устройства целиком и некоторые другие команды (например, установка системного времени).

По существу, возможны два основных сценария использования `nsgsh` в пакетном режиме:

1. В скриптах, вызываемых автоматически из текущей конфигурации устройства. В частности, такие скрипты могут вызываться при срабатывании `netring` в ту или иную сторону, при срабатывании датчиков, при получении управляющих SMS, при переходе на другой канал *u*iTCP. Как правило, их задача состоит в том, чтобы внести частные изменения в работу отдельных объектов — например, рестартовать объект (не изменяя при этом конфигурацию) или изменить состояние контроллера. Или же они ограничиваются мониторингом.

В этих случаях использовать `nsgsh` в скриптах можно, но с учётом того, что каждый конкретный вызов может получить статус `admin` или `user` случайным образом, в зависимости от наличия в этот момент

административной сессии или работы другого скрипта. Поэтому в данном случае следует использовать только команды, доступные для сессии со статусом `user`, чтобы гарантировать стабильную работу скрипта. В противном случае, как правило, скрипт работает непредсказуемо.

Для более серьёзного автоматического управления в этом случае целесообразно использовать непосредственно команды и утилиты Linux. Они работают помимо `nsgsh` и, таким образом, свободны от ограничений на число её административных сессий. Например, для манипуляций с маршрутами следует использовать утилиту `ip route`, а не `nsgsh .ip.route...`.

2. В качестве основного инструмента управления. При этом предполагается, что такие сессии запускаются эпизодически и не часто, и параллельно с ними не работают никакие другие сессии (в т.ч. и вызываемые из скриптов). Например, можно предположить, что у пользователя на ПК или на Web-сервере работает некое гипотетическое управляющее приложение, которое в конечном счете транслирует команды конфигурирования на устройство NSG по SSH в виде

```
ssh root@ip.address.of.nsg "nsgsh $COMMAND"
```

В этом случае можно использовать любые команды `nsgsh` и устанавливать значения любым параметрам.

В более сложных случаях можно начать скрипт с того, чтобы принудительно снять сессию со статусом `admin`, если таковая имеется в данный момент, и гарантировать исполнение последующих команд `nsgsh` :

```
nsgsh -q .system.sessions.close.admin; nsgsh -q команда команда ...
```

Необходимо, однако, понимать, что это очень тяжёлый и обоюдоострый инструмент, и администратор должен осознавать все возможные побочные эффекты его применения и брать на себя ответственность за это. Если, например, в момент запуска скрипта в системе работает администратор, то сделанные им и не сохранённые изменения будут утрачены. Изменения, применённые и не сохранённые, останутся в силе и отменить их можно будет только перезагрузкой устройства. Корректного решения, безболезненного для всех сторон, задача об управлении в 4 руки, увы, не имеет.

Ограничения на исполнение отдельных команд. Некоторые команды (например, создание резервной копии конфигурации и её выгрузка на удалённый сервер) требуют значительного времени для исполнения и по этой причине их исполнение в пакетном режиме запрещено. При попытке выполнить такую команду выводится соответствующее диагностическое сообщение.

Обойти это ограничение, будучи твёрдо уверенным в своих действиях, можно путём принудительного вызова `nsgsh` в интерактивном режиме, например:

```
nsgsh -i system.configurations.etc-store=file:///tmp/backup.tgz _quit
```

однако в этом случае автор скрипта принимает на себя все возможные последствия в виде, например, потери отзывчивости системы на время исполнения команды.

Команды с опциональным параметром. Для некоторых команд предусмотрен ввод параметра, но он не является обязательным — например, для рестарта порта 3G с явным выбором той либо другой SIM-карты, либо с текущей в зависимости от настроек самого порта. При вызове таких команд в пакетном режиме необходимо указать параметр явным образом — либо одно из возможных значений (например, `main`, `aux`), либо ключевое слово `nil` для того, чтобы выполнить команду с пустым значением параметра.

§7.3.3. Исполнение сценариев по результатам тестов (*ping* и др.)

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 из 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* продолжают 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:

```
NSG_LAST_STATE со значением START, UP либо DOWN — текущее состояние теста
NET_PING_EVENT со значением FAILURE либо RESTORE — событие (переход из одного состояния в
другое), вызвавшее исполнение скрипта
NET_PING_DESTINATION_IP
NET_PING_SOURCE_IP
```

Помимо *ping*, в настройках `netping` могут быть использованы другие команды для регулярной проверки какого-либо состояния, указываемая в параметре `test-script`. Подробно о настройке `netping` см. [Часть 4](#).

§7.3.4. Исполнение сценариев по событию и по заданному расписанию

Любой скрипт можно назначить в качестве действия, вызываемого в качестве реакции на некоторое событие. NSG Linux 2.0 имеет развитые средства для мониторинга и обработки событий от различных системных объектов (например, интерфейсов — переход в UP или DOWN) и внешних датчиков. Подробно о механизме генерации и обработки событий см. [Часть 4](#).

В частности, в качестве события, приводящего к исполнению скрипта, может использоваться срабатывание таймера (через заданные промежутки времени) или планировщика событий (в заданное время суток/недели/месяца и т.п.). Не запрещается также организовывать исполнение скриптов непосредственно в ОС Linux с помощью стандартного `crond`.

Для периодического исполнения команд можно также использовать скрипт в виде бесконечного цикла, содержащий внутри себя команду `sleep`, например:

```
#!/bin/sh
(
while [ 1 -ne 0 ]; do
    TEMP=$(nsgow .port.1-wire -d 1046FE6201080072)
    if echo $TEMP | grep -v "t = 2"; then
        at2 sms -p .port.edge +79012345678 "$(hostname) $TEMP"
    fi
    sleep 3600
done >/dev/null 2>&1
)&
```

Данный скрипт раз в час проверяет показания температурного датчика, подключённого к порту 1-wire. Если в ответе не содержится подстрока `t = 2`, т.е. температура ниже 20.0°C или выше 29.9°C (исключительно), то отправляет SMS.

§7.3.5. Особенности использования сценариев для рестарта

Если какой-либо из механизмов исполнения скриптов (например, *netping*) используется для рестарта сетевого интерфейса или всего устройства, то следует обратить внимание на следующие особенности:

1. Необходимо следить, чтобы время срабатывания этого механизма было гарантированно больше, чем время, необходимое для начального приведения системы в рабочее состояние (инициализации интерфейса, установления соединения). В противном случае рестарт будет происходить бесконечно. Рекомендуется, чтобы они отличались не менее чем в 2–3 раза.
2. Для перезагрузки устройства в случае возникновения нештатных ситуаций (в штатных оно не должно перезагружаться вообще никогда) рекомендуется вместо стандартной команды `reboot` использовать `nsgreboot`. Этот скрипт перед выполнением `reboot` записывает в файл `/etc/postmortem.dump` доступную отладочную информацию (журналы, `dmesg`, `ps`, `top` и т.п.), которая может быть полезна для выяснения источника проблемы.

§7.3.6. Исполнение скриптов в качестве демонов

Любой скрипт может быть запущен в NSG Linux 2.0 в качестве демона, т.е. системной службы. В этом случае система сама следит за работой скрипта, запускает его после старта, перезапускает, если он по какой-то причине прекратит работу, и т.п. Управление демонами производится в узле `.services.daemons` (подробнее см. [Часть 4](#)).

Нижеприведённый пример контролирует наличие напряжения питания 220В в некоторой розетке при помощи внешнего датчика IC-2di-220 с периодичностью в 3 секунды:

```
services
: daemons
:: check_220
::: adm-state = "up"
::: command = "while [ 1 -ne 0 ]; do \n
    if nsgsh -q .port.1-wire.device.swt2-3A460B01000000A7.circuit.A.show|grep open; then\n
        nsgsh -q .tools.led.l1.red.on;\n
        nsgsh -q .tools.led.l1.green.off;\n
    else\n
        nsgsh -q .tools.led.l1.red.off;\n
        nsgsh -q .tools.led.l1.green.on;\n
    fi;\n
    sleep 3;\n
done >/dev/null 2>&1"
```

ПРИМЕЧАНИЕ Тело скрипта вводится как одна строка; поскольку эта строка содержит специальную последовательность `\n` (или `\r`), то при последующем редактировании в Web-интерфейсе поле ввода преобразуется в двумерное текстовое окно. Для удобства работы можно сначала ввести в поле ввода `\n`, выйти из окна ввода, и войти в него заново.

§7.4. 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.

§7.4.1. 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.
```

§7.4.2. 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. `syslog --help` или соответствующие *man pages*. В частности, при вызове с ключом `-R` или `-r` журнал передаётся на удалённый сервер Syslog.

§7.4.3. uptime и top

Команда `uptime` выводит краткую информацию о времени работы системы: текущее время, время непрерывной работы системы и среднюю загрузку системы за последние 1, 5, и 15 минут.

Команда `top` выводит и периодически обновляет информацию о процессах, занимающих наибольшее количество системных ресурсов. Для выхода из команды следует нажать CTRL-C.

§7.4.4. Утилиты для настройки IP

Команда `ip` — мощный и гибкий инструмент для настройки всего сетевого стека IP и Ethernet. С её помощью возможно, в частности, настраивать IP-адреса, маршруты, службу ARP и др. Рекомендуется использовать её для всех задач вместо `ifconfig`, `route` и др. утилит.

В числе других важных утилит следует упомянуть:

```
ifconfig  Просмотр состояния и настройка IP-интерфейсов  
route     Управление IP-маршрутизацией  
arp       Просмотр таблицы ARP, настройка статического ARP и ARP проху
```

Рекомендуется использовать эти команды только для просмотра состояния системы, поскольку по своим возможностям они уступают `ip`.

§7.4.5. iptables

Пакет `IPtables` предлагает широкие возможности для манипулирования IP-пакетами, включая разнообразные варианты NAT, коммутации и т.п., выходящие за рамки типовых общеупотребительных настроек. Чтобы получить полное представление о настройках данного пакета, рекомендуется ознакомиться с *man pages*. Большинство параметров командной строки `IPtables` настраиваются средствами `nsgsh` и Web-интерфейса; недостающие могут быть указаны в полях `extra` и `extra-target`.

§7.4.6. 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 параметра `options`, например:

```
port.a1.ppp.main.options = "nobsdcomp local"
```

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

По сравнению со стандартным `pppd`, в NSG Linux 2.0 используется доработанный демон, имеющий одну дополнительную опцию конфигурации:

```
passdir директория
```

В указанной директории ищутся файлы `pap-secrets` и `chap-secrets`; если она не указана, то используется стандартная директория `/etc/ppp`.

§7.4.7. ping и traceroute

Стандартные команды для анализа функционирования IP-сети с помощью пакетов ICMP Echo Request/Reply или UDP.

§7.4.8. telnet и ssh

`telnet` и `ssh` — стандартные утилиты для управления удалёнными устройствами. 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
```

Аналогично, `telnetd` и `sshd` — стандартные сервера указанных служб для доступа к устройству NSG. Функциональность обоих серверов полностью контролируется средствами `nsgsh` и Web-интерфейса.

§7.4.9. Передача двоичных файлов по ssh

Соединение SSH может использоваться не только для защиты консольного доступа к удалённой машине, но и для передачи файлов, в том числе двоичных. Для этой цели клиент SSH на любой Linux-машине (в том числе на устройстве NSG) вызывается с параметром, содержащим команду записи или чтения из файла, и объединяется в конвейер с соответствующей командой на локальной машине. Пример передачи файла на удалённую машину:

```
cat local.file.name | ssh root@123.45.67.89 "cat > remote.file.name"
```

Извлечение файла с удалённой машины:

```
ssh root@123.45.67.89 "cat remote.file.name" | cat > local.file.name
```

В обоих случаях пароль для входа на удалённую машину будет запрошен в интерактивном режиме. Для выполнения команд необходимы соответствующие права доступа к используемым директориям. Рекомендуется помещать все принимаемые и передаваемые файлы (особенно архивы и самоинсталлирующиеся плагины NSG) в директорию `/tmp` и уже из неё распаковывать в нужную директорию.

Аналогичные средства передачи бинарных файлов имеются в некоторых SSH-клиентах для Windows. В общем случае, можно использовать локально доступное устройство с NSG Linux, чтобы поместить на него нужный файл (например, по TFTP с администраторского компьютера) и затем с него передать по SSH на удалённое устройство.

§7.4.10. Трассировка трафика — tcpdump

Для трассировки трафика и отладки проблемных соединений в состав программного обеспечения входит утилита tcpdump.

Из командной оболочки Linux tcpdump запускается следующей командой:

```
tcpdump [-v] -i ip-интерфейс
```

где обязательным параметром является имя трассируемого объекта (sN, ethN, ethN.M и т.п.).

При необходимости могут использоваться дополнительные опции, в частности:

- x или -X Вывод пакетов в шестнадцатеричном 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.

выражение Набор критериев для отбора интересующих пакетов; пакеты, не удовлетворяющие этим критериям, в трассу не выводятся.

Формат выражений и остальные опции можно посмотреть командой tcpdump --help или в *man pages* по tcpdump.

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
```

Аналогичным образом можно использовать tcpdump из Web-интерфейса или nsgsh (см. [Часть 4](#)).

§7.5. Доработанные и фирменные утилиты NSG Linux

§7.5.1. Демон конфигурации — nsgconfd

NSG Configuration Daemon — основной механизм для управления конфигурацией NSG Linux 2.0. Он запускается при старте системы, принимает команды от текстового (nsgsh) и графического (Web-сервер) интерфейсов пользователя, генерирует "на лету" необходимые файлы конфигурации, перезапускает все другие процессы в системе. В отсутствие этого демона устройство NSG представляет собой, за отдельными исключениями (*u*TCP, GRE), обычную Linux-систему с несколько доработанными пакетами, и может настраиваться обычным образом, с помощью конфигурационных файлов и скриптов.

Журнал работы демона сохраняется в файле /var/log/nsgconfd.log и может быть просмотрен средствами nsgsh, Web-интерфейса (.system.log.show) или обычными средствами ash.

§7.5.2. Утилита конфигурации — nsgsh

Консольная оболочка nsgsh представляет собой *front-end* к демону nsgconfd, предназначенный для управления устройством в режиме командной строки. nsgsh устанавливается в качестве единственно доступной командной оболочки для пользователя nsg и для всех дополнительных пользователей при подключении через Telnet, SSH или консольный порт. Интерактивная работа пользователя в nsgsh подробно описана в [Части 1](#).

nsgsh может быть запущена из ash в интерактивном режиме (без параметров, либо с ключом -i) или в пакетном режиме. При запуске в пакетном режиме операндами команды являются узлы меню, вплоть до значения конечного параметра. Использование nsgsh в скриптах описано в §7.3.2.

Помимо команд, выводимых подсказкой <Enter> в каждом узле, nsgsh имеет также две специальные команды. Они недоступны для работы в интерактивном режиме, но могут быть полезны в скриптах:

.system.get-iface-name=*имя_порта*

Вывод информации об имени IP-интерфейса, соответствующего указанному порту. Как правило, эти имена совпадают, но в некоторых важных случаях (PPP, WLAN) имена назначаются системой динамически и априори неизвестны.

.port.*имя*.get-port-parms

Только для портов, представляющих собой символьное устройство: вывод информации о физическом интерфейсе данного порта. Выводит тип интерфейса, системное имя (в терминах Linux) основного устройства для передачи данных, вспомогательного устройства для управления, режим обработчиков SMS и т.п., в зависимости от типа порта.

§7.5.3. Генератор тестового трафика — fox

Утилита fox предназначена для генерации тестового трафика. Она ожидает соединения на указанном порту TCP и при подключении к этому порту начинает слать по TCP-соединению строки вида:

```
000000098 The quick brown fox jumps over the lazy dog. Thu Dec 18 17:43:41 2008
000000099 The quick brown fox jumps over the lazy dog. Thu Dec 18 17:43:42 2008
000000100 The quick brown fox jumps over the lazy dog. Thu Dec 18 17:43:43 2008
.....
```

Запускать fox следует из отдельного сеанса telnet, из файла inittab или из стартовых скриптов. Для завершения программы в первом случае следует нажать CTRL-C. Параметры и опции:

- i *sec.cc* Интервал посылки тестовых строк, в секундах и сотых долях секунды (по умолчанию 1.00).
- c *NNNNN* Число посылаемых строк. Неограниченная активность программы не разрешена, но при необходимости можно просто установить какое-либо больше число. Значение по умолчанию — 999999999.
- host=*ip-адрес* | *
IP-адрес, на котором fox ожидает соединения. Адрес должен принадлежать одному из IP-интерфейсов данного устройства NSG. При значении * (значение по умолчанию) fox принимает запросы на установление TCP-соединений по всем адресам, назначенным данному устройству.
- port=*TCP_порт*
Номер порта TCP, на котором fox ожидает соединения. Значение по умолчанию — 50001.
- help Вывод встроенной подсказки.

§7.5.4. 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 асинхронного порта — физического RS-232 (фиксированного или сменного), внутреннего для отдельных типов модулей (IM-V34, устаревшие типы сотовых модулей IM-xxx), или эмулируемого на внутренней шине USB (современные типы сотовых модулей UM-xxx, UIM-xxx). Может указываться как имя интерфейса в дереве конфигурации NSG Linux 2.0 (напр., .port.edge), так и название устройства в терминах Linux (например, /dev/usb/acm/3). В последнем случае точное имя порта зависит от конкретной модели шасси и сменного модуля; узнать его можно командой nsgsh port.имя.get-port-parms .
- 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*
Спецсимвол для посылки сигнала BREAK; по умолчанию не установлен.
- nohwfc* Отключить аппаратное управление потоком (по умолчанию — включено).
- swfc* Включить программное управление потоком.
- nohupcl* При выходе из программы сохранять состояние сигналов DTR, RTS интерфейса неизменным.
- exit символ*
Спецсимвол для выхода из программы в терминальном режиме; по умолчанию не установлен.
- h, --help* Вывод встроенной справки.

ВНИМАНИЕ При работе с сотовыми модулями UM-xxx, UIM-xxx nsgcu работает только с указанным интерфейсом, используемым для передачи данных. Управление питанием модуля и выбором SIM-карты производится сигналами DTR, RTS вспомогательного асинхронного интерфейса или контактами GPIO, в зависимости от модели. Для управления вспомогательным интерфейсом (NSG-700) следует запустить вторую копию nsgcu в отдельной сессии. В обоих случаях, при подключению к порту штатными средствами NSG Linux 2.0 (raw-access, reverse-telnet и т.п.) управление модулем включается автоматически, поэтому рекомендуется пользоваться именно этими средствами.

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 соединение в порту.
- , 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.
- M Вывести состояние сигналов DCD и CTS.
- ? Вывести список возможных escape-последовательностей на экран.

§7.5.5. Монитор состояния сигнала DCD — dcdstarter

Утилита `dcdstarter` осуществляет контроль за состоянием входного сигнала DCD асинхронного интерфейса. Она же используется для инкапсуляции неструктурированного асинхронного потока данных напрямую в протокол TCP (Raw TCP).

Формат команды для запуска программы:

```
dcdstarter [опция значение [опция значение ...]] порт
```

Параметры и опции:

- порт* Имя наблюдаемого физического интерфейса или разъема расширения (например, `/dev/ttyS2`). Точное имя порта в Linux зависит от конкретной модели устройства и сменного модуля; узнать его можно командой `nsqsh порт.имя.get-port-parms`.
- s скорость* Скорость в порту. Значение по умолчанию 115200 бит/с.
- t формат* Формат асинхронной посылки в порту. Значение по умолчанию 8N1.
- d* Запуск в режиме демона.
- a* Запуск в режиме сервера Raw TCP (порт не инициирует соединения с удалённым сервером, а ожидает запросов от удалённого клиента.)
- i ip-адрес* Адрес сервера Raw TCP. Для порта, работающего в режиме клиента, данный параметр указывает адрес удалённого сервера. Для порта, работающего в режиме сервера, данный параметр позволяет выбрать конкретный адрес, на котором он будет ожидать соединения, из числа всех IP-адресов, принадлежащих интерфейсам данного устройства. Значение по умолчанию — 127.0.0.1.
- p порт* Номер порта TCP, на котором производится соединение Raw TCP. Значение по умолчанию — 50002. Параметр относится как к режиму клиента, так и к режиму сервера.
- k мсек* Период опроса входного сигнала DCD, в миллисекундах. Определяет скорость реакции порта на падение или подъём сигнала DCD асинхронного интерфейса. При поднятии сигнала DCD порт-клиент устанавливает соединение с сервером. Порт-сервер открывает TCP-сокеты и начинает ждать входящего соединения, но при этом не контролирует последующие изменения DCD. Если в момент поступления соединения DCD окажется опущенным, то соединение будет установлено и затем немедленно разорвано. При падении сигнала DCD оба порта (и клиент, и сервер) разрывают соединение, если оно в этот момент существует, со своей стороны. При значении по опрос DCD не производится, сигнал всегда считается поднятым, т.е. порт-клиент пытается поддерживать TCP-соединение постоянно, а порт-сервер постоянно готов к приёму входящего соединения. Значение по умолчанию — 1000 мс.
- ПРИМЕЧАНИЕ** Опрос сигнала DCD производится периодически, поэтому кратковременное изменение сигнала на противоположный и обратно может быть не замечено. Для гарантированного срабатывания сигнал DCD должен находиться в новом состоянии в течение времени не меньшего, чем установлено параметром `-k`.
- n* Отключение слежения за сигналом DCD. Сигнал считается поднятым всегда, т.е. порт-клиент Raw TCP пытается поддерживать TCP-соединение постоянно, а порт-сервер постоянно готов к приёму входящего соединения.
- с сек* Задержка перед попыткой повторного соединения, после неудачной попытки или разрыва. Относится только к режиму клиента. Значение по умолчанию — 30 сек.
- l файл* Имя файла для вывода отладочной информации. Значение по умолчанию `/var/log/aot.log`.
- r мсек* Время, на которое опускается сигнал DTR асинхронного интерфейса при разрыве TCP-соединения:
 -1 При отсутствии сетевого соединения сигнал DTR опущен постоянно.
 0 Сигнал DTR поднят постоянно, независимо от состояния соединения.
 другое Время в миллисекундах. Значение по умолчанию — 2000 мсек.
- nohwfc* Отключить аппаратное управление потоком (по умолчанию — включено).
- swfc* Включить программное управление потоком.
- v, -vv, -vvv* Степень детализации отладочной информации.
- timestamp* Включение отметок времени в отладочный файл.
- h, --help* Вывод встроенной подсказки.

§7.5.6. 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 2). 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* Системное имя сотового интерфейса или разъема расширения, в который установлен сотовый модуль (например, /dev/usb/tts/9). Точное имя порта в Linux зависит от конкретной модели устройства и сменного модуля; узнать его можно командой `nsgsh порт.имя.get-port-parms`.
- 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, но периодически nsgsms будет приостанавливать обмен данными, проверять модем на предмет наличия входящих SMS, обрабатывать их, отправлять ответы и возвращать управление pppd.
 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 AOH, то SMS-управление доступно для любых клиентов, у которых конечная 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*.
- i секунды* Период проверки SMS при работе в режиме PPP. Значение по умолчанию — 60 сек. Значение 0 запрещает прерывать работу PPP-соединения; в этом случае обработка входящих SMS производится только при рестарте порта, а отправка SMS запрещена.
- k* При запуске удалить все накопленные SMS из памяти модема и с SIM-карты, во избежание их переполнения.
- e* Завершить работу после обработки всех накопленных SMS.
- c tcp_port* Номер порта TCP, на котором обработчик SMS будет ожидать соединения от команды `at2` (см. след. параграф) в том случае, если он сам уже занимает доступный порт модема. Обработчик способен принимать команды через это TCP-соединение, передавать их в модем и ретранслировать ответы модема обратно.
- 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.

ПРИМЕЧАНИЕ Режим совмещения PPP с SMS не рекомендуется использовать на модулях IM-GPRS *h/w ver.3* (чипсеты PIML, FLYFOT).

§7.5.7. Передача SMS и исполнение AT-команд — at2

Утилита at2 выполняет заданные последовательности AT-команд для управления сотовыми модемами GSM и UMTS — в том числе и во время работы PPP-соединения. Она предназначена для использования в скриптах, контролирующих уровень сотового сигнала или отправляющих SMS по заданному расписанию или по какому-либо событию (изменению состояния интерфейсов, срабатыванию датчиков технологического контроля и т.п.).

Для современных типов сотовых интерфейсов и модулей, работающих через внутренний интерфейс USB, управление AT-командами производится через вспомогательный асинхронный порт, эмулируемый поверх USB. Служебное имя этого порта зависит от типа модуля и от разъёма, в который он установлен.

Если сотовый модуль относится к одному из устаревших типов (UIM-EDGE *h/w ver.3* или IM-xxx) и имеет единственный порт для соединения с устройством, то доступ к нему возможен только при инкапсуляции sms-handler. В этом случае утилита at2 устанавливает TCP-соединение внутри устройства к обработчику SMS (см. предыдущий параграф). Управление портом (переключение в командный режим или в режим данных) при этом возлагается на обработчик. Порту должна быть назначена инкапсуляция

```
encapsulation = "sms-handler"
```

либо

```
encapsulation = "ppp"
sms-handler.ppp-cooperation = true
```

Формат команды для запуска программы:

```
at2 [опция значение [опция значение ...]] csq
      Контроль уровня сигнала (AT+CSQ).
```

```
at2 [опция значение [опция значение ...]] rinfo
      Комплексный контроль параметров радиointерфейса: статус SIM-карты (AT+CPIN?), уровень сигнала (AT+CSQ), состояние услуг GSM и пакетной передачи данных (AT+CREG?, AT+CGREG?), выбранный оператор (AT+COPS?) и, для модулей 3G, текущий режим пакетной передачи (AT+CNSMOD?).
```

```
at2 [опция значение [опция значение ...]] sms номер_телефона текст
      Отправка SMS на указанный номер. Если текст содержит пробелы, его необходимо заключить в кавычки. Текст может также содержать вывод команд — например, $(hostname) — и переменные окружения.
```

Опции программы:

```
--port=имя или -p имя
```

Имя сотового интерфейса полностью, начиная от корня дерева конфигурации (например, .port.s1).

```
--dev=путь
```

Сетевое устройство, через которое осуществляется работа с портом, в обозначениях Linux (например, /dev/usb/lpp0). Точное имя порта в Linux зависит от конкретной модели устройства и сменного модуля; узнать его можно командой `nsgsh port.имя.get-port-parms`.

```
--tcpport 1024...65535
```

Номер порта TCP, на котором обработчик SMS ожидает соединения. (По умолчанию — 50000.)

Из трёх вышеуказанных опций в любом случае должна использоваться ровно одна. В большинстве случаев следует использовать опцию --port; параметры --dev или --tcpport в этом случае определяются автоматически в зависимости от конкретной модели устройства и сменного модуля. Сами по себе эти опции предназначены для ручной настройки новых модулей, ещё не внесённых в имеющуюся версию ПО.

```
--short      Вывод ответов модема в краткой форме для упрощения их дальнейшей обработки в скриптах.
              Например, для at2 csq будет выводиться одна цифра — уровень сигнала.
```

```
--help      Вывод встроенной справки.
```

После соединения с модемом выводятся полностью его ответы. В частности, при успешной отправке SMS выводится её порядковый номер и сообщение OK.

```
root@nsg700 root # at2 -p=.port.edge sms +79012345678 "test preved"
+CMGS: 61
OK
```

Если соединиться с модемом не удаётся в течение 10 сек. (например, модем находится в процессе перезагрузки), то не выводится ничего.

ВНИМАНИЕ Если модуль работает в режиме совмещения SMS и PPP, то исходящие SMS отправляются только в том случае, если обработчик SMS получает управление в то время, пока at2 ожидает ответа. Хранение SMS до следующего "удобного момента" не производится, поскольку целесообразность отправки такого запоздалого сообщения зависит от конкретной задачи. При необходимости пользовательский скрипт должен сам следить за результатом выполнения at2 и повторять попытки, если это имеет смысл в данной задаче.

§7.5.8. Управление по шине 1–Wire — nsgow

Для доступа к датчикам и контроллерам технологического управления, подключённым по шине 1–Wire, используется утилита nsgow. Формат команды для запуска программы:

```
nsgow [опция значение [опция значение ...]] порт
```

При этом в основной командной оболочке для данного порта может быть установлено encapsulation one-wire или encapsulation unused. Для данного типа портов это несущественно, поскольку обращения к шине (не важно, из какой оболочки) выполняются разово и не занимают порт постоянно.

Параметры и опции:

порт Имя фиксированного или сменного порта 1–Wire. Может указываться как имя интерфейса в дереве конфигурации NSG Linux 2.0 (напр., .port.1-wire), так и название устройства в терминах Linux (например, /dev/usb/tts/21). В последнем случае точное имя порта зависит от конкретной модели шасси и сменного модуля; узнать его можно командой nsgsh port.имя.get-port-parms . Для подключения шины 1–Wire может использоваться любой асинхронный порт RS–232 (фиксированный или сменный) с внешним адаптером RS–232/1–Wire, встроенный порт 1–Wire на некоторых моделях устройств, или сменный интерфейсный модуль NSG IM–1W на шасси NSG–700.

-i Вывести шестнадцатеричные идентификаторы устройств(а).

-s Вывести шестнадцатеричные идентификаторы устройств(а), и их состояния.

-k Вывести состояния устройств в краткой числовой форме.

При вызове команды без вышеперечисленных опций выводится состояние устройств(а) в текстовой форме.

Если данные опции используются без -d, выводится информация обо всех устройствах. Порядок перечисления устройств при указании -k или отсутствии опций такой же, как в остальных случаях вывода.

-d *идентификатор[:тип]*

Обратиться к конкретному устройству на шине по его шестнадцатеричному идентификатору. Параметр *тип* для большинства устройств может иметь одно из значений i (устройство ввода, *read-only*), o (гипотетическое устройство вывода, *write-only*) или b (двунаправленное устройство, *read-write*).

По умолчанию, для всех устройств предполагается тип i, поэтому для записи в устройство необходимо указать тип o или b.

Для термодатчиков допускается тип celc (по умолчанию) либо fahr, устанавливающий единицы измерения.

Поскольку команда выполняется разово, никакие установки от предыдущего вызова не сохраняются, и данную опцию необходимо указывать полностью при каждом вызове.

-A *операция*

-B *операция*

.....

-O *операция*

.....

-7 *операция*

Установить состояние для конкретных электрических цепей, подключенных к устройству. Количество и наименования выходных пар зависят от модели устройства. Допускается только в сочетании с опцией -d для устройств, доступных на запись, т.е. имеющих тип o или b. Поддерживаемые операции:

short Замкнуть цепь.

open Разомкнуть цепь.

toggle Изменить состояние цепи на противоположное.

pulse Замкнуть цепь на время, установленное опцией -r, затем снова разомкнуть её. Если в исходном состоянии цепь замкнута, она будет просто разомкнута через время -r.

drop Разомкнуть цепь на время, установленное опцией -r, затем снова замкнуть её. Если в исходном состоянии цепь разомкнута, она будет просто замкнута через время -r.

-r *мсек* Установить продолжительность промежуточного состояния для операций pulse и drop, в миллисекундах. В связи с особенностями работы шины и конкретных устройств 1–Wire, фактическое время исполнения команды может заметно варьироваться.

-h, --help Вывести справку о команде.

§7.5.9. Отправка электронных писем — `nsgsmtp`

Для отправки сообщений по электронной почте используется утилита `nsgsmtp`, которая может использоваться в любых скриптах ОС Linux или обработчиках событий. Формат команды для запуска программы:

```
nsgsmtp --from=sender@domain --rcpt=recipient@domain --server=host [опция значение [опция значение ...]]
```

Параметры и опции:

- `-f, --from=` Почтовый адрес отправителя письма. Параметр обязательный.
- `-r, --recipient=` Почтовый адрес получателя письма. Параметр обязательный.
- `--subject=` Тема письма. Параметр обязательный по существу; если не указан, письмо отправляется с заголовком "No subject" .
- `--body=` Текст письма. Параметр обязательный по существу; если не указан, письмо отправляется с текстом "No message".
- `--server=` Доменное имя или IP-адрес сервера SMTP. Параметр обязательный.
- `--ssl` Использовать безопасный режим SMTP-over-SSL.
- `--port=` Номер порта TCP, используемого по умолчанию сервером SMTP. Значение по умолчанию для обычного режима работы — 25, для безопасного — 465.
- `--user=`
- `--password=` Имя и пароль пользователя на сервере SMTP. Требуется в случае, если сервер требует аутентификации.
- `--domain=` Доменное имя устройства NSG, передаваемое в поле `helo` протокола SMTP. Требуется в случае, если почтовый сервер имеет строгие настройки для защиты от рассылки спама неавторизованными пользователями, и должно соответствовать IP-адресу устройства NSG, с которого отправляется письмо.
- `-q` Режим работы без диагностических сообщений.
- `-h, --help` Вывести справку о команде.

Строковые значения, содержащие пробелы (тема и тело письма) должны вводиться в двойных кавычках.

§7.5.10. Доступ к удалённому устройству и синхронизация конфигурации — `nsgconfsync`

Утилита `nsgconfsync` выполняет три функции, первая из которых необходима для двух других:

- Генерацию ключей для клиента SSH и передачу публичного ключа на указанный удалённый сервер SHH. Таким образом, обеспечивается последующий доступ на удалённое устройство без ручного ввода пароля для автоматизированного выполнения тех или иных операций на нём. Данная функция выполняется вручную один раз (или периодически для регулярной смены ключей) и может использоваться как с целью последующего импорта конфигурации с удалённого устройства на локальное, так и для иных задач: мониторинга распределённой системы `iitcp` с одного сервера, трансляции событий на подключённых датчиках и контроллерах, и т.п. Удалённым хостом может быть как устройство NSG, так и любое другое устройство со стандартным для Linux назначением директорий SSH.
- Чтение дерева конфигурации с удалённого хоста по SSH и его слияние с текущей конфигурацией локального устройства по установленным правилам. Предполагается, что данная функция применяется к двум однотипным и функционально идентичным устройствам NSG, например, основному и резервному серверам VPN. В этом случае администратор вносит изменения только на основном сервере (например, добавляет новых пользователей), а на резервных серверах выполняет данную процедуру и копирует на них требуемые элементы конфигурации с основного.
- Копирование заданных файлов и директорий с удалённого устройства на локальное.

Формат команды

```
nsgconfsync [username@hostname] [OPTION ...]
```

Опции для генерации и передачи ключа:

- k, --keygen Сгенерировать пару ключей RSA путем вызова команды `ssh-keygen -t rsa`. На все последующие вопросы следует отвечать нажатием клавиши `Enter`.
- a, --append Передать свой публичный ключ на указанный сервер. Для передачи ключа устанавливается соединение SSH с ручным вводом пароля, и ключ помещается в соответствующую директорию SSH на сервере.

Опции для синхронизации конфигурации:

- i, --import=NODE
Загрузить данный узел конфигурации с удалённого устройства. Допускается указание данной опции несколько раз.
- e, --exclude=NODE
Исключить данный узел из загруженной конфигурации и восстановить его исходное состояние на локальном устройстве. Допускается указание данной опции несколько раз.
- f, --file=FILE
Загрузить данный файл или директорию с удалённого устройства. Путь к файлу указывается от корня файловой системы. В последнем элементе пути (собственно имени файла или директории) допускается использование подстановочного символа `*`. Допускается указание данной опции несколько раз.
- excludefile=FILE
Исключить файл или директорию из синхронизации. В последнем элементе пути (собственно имени файла или директории) допускается использование подстановочного символа `*`. Путь к файлу указывается от корня файловой системы. Допускается указание данной опции несколько раз.

ПРИМЕЧАНИЕ Если на локальном устройстве существуют файлы с такими же путями/именами, как на мастер-устройстве, то они будут заменены. Если в локальных директориях есть файлы, которых нет на мастер-устройстве, то они останутся как есть и удалены не будут.

- script=SCRIPT
После завершения синхронизации выполнить указанный скрипт. Выполняется только при установленной опции `--write` и успешном завершении всех предыдущих операций.
- h, --help
Вывести справку о команде.
- q, --quiet
Не выводить в журнал исходную локальную конфигурацию, принятую конфигурацию и результирующую конфигурацию. Сообщения о результате синхронизации и об ошибках выводятся в любом случае.
- s, --sync
Выполнить синхронизацию.
- w, --write
Сохранить результирующую конфигурацию в энергонезависимой памяти устройства.
- apply=NODE
Применить результирующую конфигурацию для данного узла. Допускается указание данной опции несколько раз.
- force
Если в момент выполнения синхронизации на устройстве открыта сессия администратора, принудительно завершить её (возможна потеря несохранённых изменений) и запустить `nsgconfsync` с правами администратора, чтобы иметь возможность применять и сохранять новую конфигурацию. Рекомендуется при автоматическом вызове процедуры; при ручном выполнении синхронизации следует использовать опции `--write` и `--apply` в этой же строке вызова.

§7.6. Участие пользователей в развитии 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) компании 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 на устройствах расширения энергонезависимой памяти (карте SD, USB Flash или USB HDD, в зависимости от модели шасси). При этом следует иметь в виду, что любая твердотельная память предназначена, в основном, для хранения приложений. Для часто перезаписываемых пользовательских данных, таких как статистика, журналы ввода-вывода и т.п., рекомендуется использовать жесткий диск.

В отсутствие дополнительных устройств памяти приложения пользователя следует устанавливать в директорию /root — 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.

ПРИМЕЧАНИЕ При настройке инструментария разработчика необходимо обратить особое внимание на правильный выбор целевой платформы, а также формата Little/BigEndian, для конкретной модели устройства NSG. Тип процессора указан в документации на устройство. Компания NSG не оказывает консультаций (в т.ч. платных) по разработке собственных приложений пользователя. Следует руководствоваться общедоступными материалами по разработке приложений Linux для встраиваемых систем.