



**ОЛЕГ ЧЕРНЫЙ**

**АДМИНИСТРИРОВАНИЕ LINUX**

**КОНСПЕКТ ЛЕКЦИЙ**

**Одесса 2012**

■ Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава  
■ Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков



# 1. Управление программным обеспечением

В этой главе будут рассмотрены Zypper и RPM, утилиты командной строки, с помощью которых можно управлять программным обеспечением.

## 1.1 Использование Zypper

Zypper – утилита командной строки, которая используется для инсталляции, удаления и обновления программного обеспечения. В своей работе Zypper использует репозитории (хранилища) пакетов, которые находятся по определенным адресам в сети Интернет или в сети предприятия. Для инсталляции программного продукта можно указать лишь его название в аргументах утилиты Zypper, и она самостоятельно найдет инсталляционный пакет данного программного продукта, определит какие пакеты должны быть установлены дополнительно, загрузит все необходимые пакеты из сети Интернет и установит их.

### 1.1.1 Установка обновлений

Основной синтаксис Zypper:

```
zypper [global-options] command [command-options] [arguments] ...
```

Компоненты, которые указаны в квадратных скобках, не являются обязательными. Например, для установки всех необходимых обновлений достаточно воспользоваться данной командой:

```
zypper patch
```

Для того чтобы в ходе инсталляции не задавались какие-либо вопросы (выбирались ответы, которые заданы по умолчанию), можно воспользоваться дополнительной опцией `--non-interactive`:

```
zypper --non-interactive patch
```

Для установки обновлений и автоматического принятия необходимых лицензионных соглашений можно воспользоваться опцией `--auto-agree-with-licenses`:

```
zypper patch --auto-agree-with-licenses
```

Для того чтобы узнать количество доступных и еще не установленных обновлений, используется команда *patch-check*:

```
zypper patch-check
```

Для получения списка всех необходимых, но еще не установленных обновлений, используется команда *List-patches*:

```
zypper list-patches
```

Для получения списка всех обновлений, как установленных, так и не установленных в системе, используется команда *patches*:

```
zypper patches
```

Если в репозиториях хранится лишь свежая версия программного продукта, и не хранятся обновления (патчи) для него, то необходимо пользоваться командой *update*, с помощью которой осуществляется обновление программного продукта до его актуальной версии:

```
zypper update
```

Для обновления конкретного пакета используется следующая команда:

```
zypper update имя-пакета
```

Для получения списка всех пакетов, у которых появились более свежие версии, используется команда *List-updates*:

```
zypper list-updates
```

### 1.1.2 Инсталляция и удаление пакетов

Для установки пакетов используется команда *install*. В таком случае необходимо указывать в качестве аргумента имя пакета или пакетов, которые необходимо проинсталлировать в системе:



```
zypper install MozillaFirefox
```

Для инсталляции пакетов *mpLayer* и *amarok* с использованием только репозитория *factory* с выводом подробной информации о ходе инсталляции, используется следующая конструкция:

```
zypper -v install -repo factory mpLayer amarok
```

Для удаления пакетов используется команда *remove*:

```
zypper remove MozillaFirefox
```

Передать название пакета в качестве аргумента можно выполнить, используя шаблон:

```
zypper install MozillaF*
```

Можно указывать путь к требуемому пакету:

```
zypper install /tmp/install/MozillaFirefox.rpm  
zypper install http://download.opensuse.org/SUSE/MozillaFirefox-3.5.x86_64.rpm
```

В одной команде можно одновременно устанавливать и удалять пакеты, используя модификаторы «+» и «-». Например, две нижеследующие команды выполняют одно и то же действие: устанавливают пакет *emacs* и удаляют пакет *vim*:

```
zypper install emacs -vim  
zypper remove vim +emacs
```

Пакеты связаны друг с другом зависимостями. Зависимость означает то, что для корректной работы определенного программного продукта в системе должен быть установлен другой программный продукт. Для проверки того, соблюдаются ли все необходимые зависимости, в системе используется команда *verify*:

```
zypper verify
```

По результатам выполнения предыдущей команды некоторые пакеты, для выполнения условий зависимости, получают статус «рекомендованных». Для инсталляции таких пакетов используется нижеследующая команда:

```
zypper install-new-recommends
```

### 1.1.3 Управление репозиториями

Для получения списка всех известных в системе репозиториях используется следующая команда:

```
zypper repos
```

Для добавления репозитория используется следующая команда:

```
zypper addrepo URI Alias
```

В качестве *URI* можно использовать адрес в Сети Интернет, адрес сетевого ресурса, локальный каталог, CD или DVD. *Alias* – любое уникальное имя, которое будет служить уникальным идентификатором репозитория.

Для удаления N-го по счету репозитория из списка репозиториев используется следующая команда:

```
zypper removerepo N
```

### 1.1.4 Поиск пакетов в репозиториях

Zypper предоставляет довольно широкие методы поиска пакетов в репозиториях. Осуществим простой поиск по имени:

```
zypper search firefox
```

А теперь поиск с использованием шаблонов и специальных символов «?» (один любой символ) и «\*» (любое количество любых символов):

```
zypper search f?ref*
```

Поиск не только по именам пакетов, но и в полях с дополнительной информацией:

```
zypper search -d fire
```

Отобразить только еще неустановленные пакеты:

```
zypper search -u firefox
```

## 1.2 RPM Package Manager

RPM Package Manager традиционное средство управления пакетами, используемое во всех RPM-based дистрибутивах Linux. Инсталляционные RPM архивы (пакеты) представляют собой файлы, которые содержат программные файлы и дополнительную информацию, используемую при установке пакета в системе. Обычно такие пакеты имеют расширение *.rpm*.

### 1.2.1 Управление пакетами

Для проверки целостности пакета, подлинности его авторства, следует пользоваться следующие командой:

```
rpm --checksig package-1.2.3.rpm
```

Таким образом можно узнать кто является поставщиком пакета, был ли он выпущен доверенным источником, и принять решение, стоит ли инсталлировать данный пакет в системе.

Для инсталляции пакета можно воспользоваться простейшей командой:

```
rpm -i package-1.2.3.rpm
```

Пакет будет установлен только при условии выполнения всех необходимых зависимостей. База данных RPM пакетов должна быть построена таким образом, что какой-то определенный файл может принадлежать только одному определенному пакету. Очевидно, что такой подход имеет как свои преимущества, так и недостатки.

Для обновления пакета, что подразумевает его удаление и установку более свежей версии, используются следующий команды:

```
rpm -U package-1.2.3.rpm
rpm --upgrade package-1.2.3.rpm
rpm -F package-1.2.3.rpm
rpm --freshen package-1.2.3.rpm
```

Отличие ключа *-U* от ключа *-F* состоит в том, что опция *--upgrade* установит пакет в любом случае, независимо от того, установлена ли его предыдущая версия в системе, или нет.



Для удаления пакета используется данная команда:

```
rpm -e package
```

Аналогично инсталляции, пакет может быть удален только при условии выполнения всех необходимых зависимостей. Однако можно пренебречь данным требованием, и игнорировать все зависимости, настаивая на том, что необходимо произвести удаление или установку пакета, невзирая на конфликты.

```
rpm -e --nodeps package
```

Для выполнения запросов о пакете, используется ключ *-q*. Если пакет установлен в системе, то о нем можно получить информацию, используя ключ *-i*. Если пакет еще не установлен, то необходимо использовать ключ *-r*. Для получения списка файлов, которые содержатся в пакете, используется ключ *-l*. Например:

```
rpm -qipl /tmp/install/package-1.2.3.rpm
rpm -qil package
```

Отличие двух выше указанных команд в том, что первая из них обращается за искомой информацией непосредственно в инсталляционный пакет, а вторая к базе пакетов, которые уже проинсталлированы в системе. Файлы базы данных RPM находятся в каталоге */var/lib/rpm*.

### 1.3 Задания для самопроверки

1. Установите программу Midnight Commander используя утилиты командной строки.
2. Определите, какое количество файлов было скопировано в каталог */usr/bin* в ходе инсталляции Midnight Commander.
3. Изучите список репозиториев, к которым обращается ваша система. На сайте [http://en.opensuse.org/Package repositories](http://en.opensuse.org/Package_repositories) изучите список доступных репозиториев. Добавьте в свой локальный список какой-либо репозиторий пакетов, который, по вашему мнению, мог бы стать полезным для вас.
4. Определите, доступны ли какие-либо обновления безопасности для вашей системы. Установите необходимые обновления, чтобы ваша система была защищена от потенциальных уязвимостей.
5. С помощью утилиты Midnight Commander зайдите в какой-либо *.rpm* пакет (клавишей Enter) и проанализируйте представленную информацию.

## 2. Загрузка Linux

В этой главе будут рассмотрены вопросы, связанные с процессом загрузки системы, начиная с этапа инициализации аппаратного обеспечения средствами BIOS и заканчивая изучением программы *init*, отвечающей за инициализацию системы в нужном для администратора направлении.

### 2.1 BIOS

После включения компьютера, BIOS (Basic Input Output System) инициализирует экран и клавиатуру, тестирует основную память. Различную важную информацию, например, о текущей дате и времени, загружается из памяти, под названием CMOS. Именно она содержит настройки, которые администратор сохраняет в программе BIOS Setup.

Когда первый жесткий диск и его геометрия распознаны, управление по загрузке системы передается загрузчику операционной системы, который, как правило, находится в самом первом секторе (нулевом) жесткого диска. Возможна загрузка и по сети, если BIOS вашей системы поддерживает такой вариант загрузки.

### 2.2 Загрузчик

В нулевом секторе жесткого диска хранится таблица разделов и код загрузчика операционной системы. Эти первые 512 байт жесткого диска часто называют Главной Загрузочной Записью (Master Boot Record, MBR). Структура *MBR* не зависит от того, какая используется операционная система. 446 байт зарезервировано за программным кодом. Как правило, это часть загрузчика операционной системы. Оставшиеся 64 байта используется для хранения таблицы разделов, в которой может быть не более четырех записей. Один из этих разделов может быть отмечен как «активный», наличие активного раздела является обязательным условием для некоторых операционных систем. Последние два байта MBR должны содержать обязательное значение 55AA. В случае если это значение будет другим, загрузка системы не будет успешной.

Первый сектор раздела называется загрузочным. В 512 байтах данного сектора содержится код, который используется для загрузки операционной системы, установленной, соответственно, на данном разделе. Однако это не относится к Linux системам, загрузочный раздел Linux систем, как правило, пуст.

После того, как MBR помещается в основную память, следующая часть процесса загрузки переходит под управление загрузчика, команды которого последовательно выполняются процессором.

Для сетевой загрузки, в роли загрузчика выступает BIOS. BIOS загружает образ операционной системы со специального сетевого сервера и запускает систему. В таком случае процесс загрузки полностью независим от локального жесткого диска.

### 2.2.1 Загрузчик GRUB

GRUB (Grand Unified Bootloader) является основным загрузчиком для современных Linux систем. Работа GRUB состоит из двух основных стадий (*stages*). Первая стадия (*stage 1*) содержит 512 байт кода и ее задача лишь в запуске второй стадии (*stage 2*). *Stage 2* содержит основную часть загрузчика. В некоторых системах может присутствовать промежуточная стадия – *stage 1.5*. Ее задача состоит в поиске и обнаружении местоположения *stage 2* на соответствующей файловой системе.

*Stage 2* может обращаться к различным файловым системам: Ext2, Ext3, ReiserFS, Minix, DOS FAT, XFS, UFS и другие. Конфигурация GRUB хранится в основных файлах:

#### /boot/grub/menu.lst

В этом файле содержится информация обо всех операционных системах, которые могут быть загружены с помощью GRUB. Без информации в данном файле остается возможность запуска операционных систем, используя командную строку, встроенную в загрузчик GRUB.

#### /boot/grub/device.map

Этот файл производит трансляцию имен из BIOS и GRUB нотаций в наименования устройств, принятых в Linux.

#### /etc/grub.conf

Этот файл содержит команды и параметры, необходимые для корректной установки загрузчика GRUB.

#### /etc/sysconfig/bootloader

Для SUSE Linux характерно наличие конфигурационных файлов, находящихся в каталоге */etc/sysconfig*, для различных утилит и служб. В данные файлы сохраняет

■ Киев	■ Днепропетровск	■ Львов	■ Ровно	■ Мариуполь	■ Полтава
■ Одесса	■ Донецк	■ Николаев	■ Запорожье	■ Луганск	■ Харьков



свои изменения утилита YAST, с помощью которой в SUSE Linux можно администрировать систему в графической оболочке. Кроме того, данный файл автоматически редактируется при инсталляции нового ядра.

### 2.2.1.1. /boot/grub/menu.lst

При каждой загрузке системы GRUB загружает меню выбора операционных систем из конфигурационного файла, находящегося на определенной файловой системе. Рассмотрим пример конфигурационного файла.

```
gfxmenu (hd0,4)/boot/message
color white/blue black/light-gray
default 0
timeout 8

title Linux
root (hd0,4)
kernel /boot/vmlinuz root=/dev/sda7 vga=791
initrd /boot/initrd

title Windows
rootnoverify (hd0,0)
chainloader +1

title floppy
rootnoverify (fd0)
chainloader +1

title failsafe
root (hd0,4)
kernel /boot/vmlinuz root=/dev/sda7 ide=nodma apm=off acpi=off \
vga=normal nosmp maxcpus=0 noresume
initrd /boot/initrd
```

Первый блок определяет графическое оформление меню загрузки:

```
gfxmenu (hd0,4)/boot/message
фоновый рисунок хранится в каталоге /boot/message раздела /dev/sda5

color white/blue black/light-gray
цветовая схема определяет оформление пунктов меню в текстовом режиме
работы.
```

- Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава
- Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков





```
default 0
```

по умолчанию будет загружена операционная система, находящаяся в меню загрузке первой (нумерация начинается с нуля).

```
timeout 8
```

если в течение 8 секунд после появления меню загрузки пользователь не вводил ничего с клавиатуры, автоматически запускается операционная система, указанная в опции *default*.

Следующим, и самым большим блоком, является список разнообразных операционных систем. Для идентификации, пункт меню операционной системы начинается со своей собственной опции *title*. Текст, указанный после этой опции будет выступать в роли названия соответствующего пункта меню.

```
title Linux
    root (hd0,4)
    kernel /boot/vmlinuz root=/dev/sda7 vga=791
    initrd /boot/initrd
```

Опция *root* указывает на то, что файловая система, на которой хранится ядро операционной системы, находится на первом логическом диске (логический диск создается в extended разделе жесткого диска) первого жесткого диска.

Опция *kernel* указывает полный адрес файла ядра операционной системы и параметры, которые будут переданы ядру для корректной загрузки. В частности, параметр *root* – раздел, на котором находится корневая файловая система, *vga* – разрешение экрана, которое будет использовано.

Вторая запись *title* отвечает за загрузки Windows:

```
title Windows
    rootnoverify (hd0,0)
    chainloader +1
```

Windows будет загружена с первого раздела первого жесткого диска. Опция *chainLoader +1* используется для того, чтобы запустить содержимое первого сектора указанного раздела.



Для запуска системы с флоппи-диска, без редактирования порядка загрузки в BIOS, используется пункт меню *floppy*. Пункт меню *failsafe* используется для запуска системы в «безопасном» режиме. В режиме, в котором выбраны такие параметры ядра, при которых вероятность неудачной загрузки системы значительно снижена.

### 2.2.1.2. /boot/grub/device.map

Данный файл служит для сопоставления имен устройств, принятых в соглашении об именовании в GRUB и в Linux. Порядок устройств, представленных в данном файле, соответствует порядку устройств, который выбран нами в качестве загрузочных в BIOS Setup. Например:

```
(hd0)      /dev/disk/by-id/Disk1_ID
(hd1)      /dev/disk/by-id/Disk2_ID
(fd0)      /dev/fd0
```

### 2.2.1.3. /etc/grub.conf

Файл содержит команды, опции и параметры, необходимые для корректной инсталляции GRUB в соответствующий сектор жесткого диска. Например:

```
setup --stage2=/boot/grub/stage2 --force-lba (hd0,1) (hd0,1)
quit
```

Данной командой производится установка GRUB на второй раздел первого жесткого диска, используя загрузочные образы, находящиеся на том же разделе. Опция *--stage2=/boot/grub/stage2* необходима для инсталляции *stage2* образа со смонтированной файловой системой. В некоторых BIOS, некорректно работает LBA режим, и опция *--force-lba*, позволяет игнорировать данную проблему.

### 2.2.1.4. Установка пароля на загрузку

GRUB получает доступ к файловой системе еще до того, как загружается операционная система. Командная строка, встроенная в GRUB, имеет достаточно функциональных возможностей для того, чтобы обращаться к содержимому файловой системы без

- █ Киев
- █ Днепропетровск
- █ Львов
- █ Ровно
- █ Мариуполь
- █ Полтава
- █ Одесса
- █ Донецк
- █ Николаев
- █ Запорожье
- █ Луганск
- █ Харьков





ввода пароля каких-либо пользователей. Необходимо устанавливать парольную защиту на загрузку, в противном случае неизбежны нарушения в безопасности системы, вплоть до потери пароля пользователя *root*. Для установки такой защиты нужно выполнить следующую последовательность действий:

- сгенерируйте зашифрованный пароль, используя *grub-md5-crypt*:

```
# grub-md5-crypt
Password: ****
Retype password: ****
Encrypted: $1$oe5Sq/$KXWwDz3SNaTs7JFw9MHvE.
```

- вставьте сгенерированную строку с паролем в глобальной секции файла */boot/grub/menu.lst*:

```
gfxmenu (hd0,4)/boot/message
color white/blue black/light-gray
default 0
timeout 8
password --md5 $1$oe5Sq/$KXWwDz3SNaTs7JFw9MHvE.
```

- чтобы запретить пользователям, не знающим специальный пароль, запускать определенные операционные системы, добавьте опцию *lock* в секции *title* соответствующих пунктов меню:

```
title Linux
root (hd0,4)
kernel /boot/vmlinuz root=/dev/sda7 vga=791
initrd /boot/initrd
lock
```

Пароли для запуска операционных систем могут отличаться. Сгенерируйте с помощью *grub-md5-crypt* необходимые пароли и вставьте их в секции *title* аналогично тому, как мы это делали в глобальной секции файла *menu.lst*:

```
title Linux
...
password --md5 $1$oe5Sq/$KXWwDz3SNaTs7JFw9MHvE
...
title Windows
...
password --md5 $1$ajlaI/$KJkLaLiUGKLkLJ98JLLmL
```





## 2.3 Init

Прежде чем корневая файловая система может быть смонтирована для работы, ядру необходимы соответствующие драйверы для доступа к устройству, на котором расположена корневая файловая система. Это могут быть драйверы каких-либо дисковых накопителей или драйверы сетевых устройств для доступа к сетевой файловой системе (Network File System, NFS). Необходимые драйверы могут быть загружены программой *init*, которая находится в *initramfs*.

Основная задача программы *init*, которая находится в *initramfs*, состоит в том, чтобы подготовить монтирование корневой системы и обеспечение доступа к ней. В зависимости от системы, *init* может выполнять следующие действия: загружать модули ядра, создавать специальные файлы устройств в каталоге */dev*, управлять настройками RAID и LVM томов, управлять сетевыми настройками.

После выполнения начальной инициализации системы и монтирования корневой файловой системы, с жесткого диска запускается «основная» версия программы *init*. Идентификатор процесса (*process ID*) данной программы – 1. Таким образом, это самый главный процесс в операционной системе, чью работу невозможно остановить стандартными средствами, доступными администратору системы.

*Init* руководствуется в своей работе конфигурационным файлом */etc/inittab*, в котором определены уровни исполнения системы (*runLevels*). В этом же файле могут быть определены службы и программы, которые запускаются на определенных уровнях исполнения. Когда процесс начальной инициализации системы завершен, *init*, основываясь на */etc/inittab*, принимает решение о том, на какой уровень исполнения должна быть переведена система.

### 2.3.1 Уровни исполнения

Уровни исполнения определяют порядок старта системы и то, какие службы доступны в запущенной системе. Параметр *initdefault* файла */etc/inittab* определяет уровень исполнения, на который будет переведена система при ее старте. Номер уровня можно указать и перед загрузкой. Для этого нужно передать команде *kernel* загрузчика GRUB параметр, в виде желаемого номера, например «5». Этот параметр будет обработан программой *init* при загрузке системы. По умолчанию, в современных системах, как





правило, используется 5 уровней исполнения, позволяющий пользователям работать в графическом интерфейсе системы.

Уровень исполнения	Описание
0	Остановка системы
5 или 1	Однопользовательский режим
2	Локальный однопользовательский режим без поддержки сетевых подключений
3	Полнофункциональный многопользовательский режим без графической оболочки
4	Определяемый администратором уровень.
5	Полнофункциональный многопользовательский режим с графической оболочкой
6	Перезагрузка системы

Для изменения уровня исполнения во время работы системы, используется команда *telinit*. В аргументах данной команды нужно указать желаемый номер уровня исполнения. Например:

```
telinit 5
```

При переходе с 3 уровня исполнения на 5 уровень исполнения, производится следующий порядок действий:

1. Администратор (root) вводит команду *telinit 5*
2. *Init* проверяет какой уровень исполнения является текущим и запускает команду */etc/init.d/rc 5*, где 5 – номер нового уровня исполнения
3. *Init* запускает скрипты, которые останавливают те службы, для которых не предусмотрен запуск на 5 уровне исполнения. Это скрипты, которые находятся в каталоге «старого» уровня */etc/init.d/rc3.d*. Имя таких скриптов начинается с латинской буквы «K» (kill).
4. *Init* запускает скрипты из каталога «нового» уровня - */etc/init.d/rc5.d*. Выполняются те скрипты, чье имя начинается с «S» (start).

В случае если производится переход на тот же уровень исполнения, *init* только проверяет и применяет изменения, сделанные администратором в файле */etc/inittab*. Точно такую же функциональность предоставляет команда *telinit q*.



## 2.3.2 Init скрипты

Существует два типа скриптов, которые находятся в каталоге `/etc/init.d`. Первый тип - скрипты, которые запускаются непосредственно программой `init`. Это происходит только при старте системы или при незамедлительном завершении работы системы (при нажатии Ctrl+Alt+Delete в консоли или при получении сигнала от источника бесперебойного питания). Выполнение таких скриптов описано непосредственно в файле `/etc/inittab`. Второй тип - скрипты, которые косвенно запускаются программой `init`. Такие скрипты запускаются при переходе на определенный уровень исполнения и для их запуска всегда вызывается *master script* - `/etc/init.d/rc`, что гарантирует правильный порядок выполнения скриптов, в последовательности, которая указана администратором.

Все скрипты находятся в каталоге `/etc/init.d`. Скрипты, которые запускаются в ходе загрузки системы, до ее перехода на какой-либо уровень исполнения, вызываются через мягкие ссылки, которые созданы в каталоге `/etc/init.d/boot.d`. Скрипты, которые запускаются при переходе на какой-либо уровень исполнения, вызываются через мягкие ссылки, которые созданы в каталогах `/etc/init.d/rcN.d`, где N - номер уровня исполнения. Так как скрипт может отвечать как за запуск службы, так и за ее остановку, он должен принимать такие параметры как `stop` и `restart`. Кроме того, поддерживается еще ряд параметров:

Параметр	Описание
<code>start</code>	Запуск службы
<code>stop</code>	Остановка службы
<code>restart</code>	Перезапуск службы. Если служба не запущена – запуск.
<code>reload</code>	Перечитать конфигурацию службы без ее перезапуска
<code>force-reload</code>	Перечитать конфигурацию службы, если она поддерживает такую функцию. В противном случае, выполнить перезапуск службы
<code>status</code>	Отобразить текущее состояние службы

Для создания собственного скрипта необходимо использовать шаблон `/etc/init.d/skeleton`. Сделайте копию данного файла, присвойте ему необходимое

- Киев
- Днепропетровск
- Львов
- Ровно
- Мариуполь
- Полтава
- Одесса
- Донецк
- Николаев
- Запорожье
- Луганск
- Харьков





имя и отредактируйте его содержимое. Блок *INIT INFO* такого скрипта обязательно должен быть заполнен. Например:

```
### BEGIN INIT INFO
# Provides:           STEP
# Required-Start:    $sysLog
# Required-Stop:      $sysLog
# Default-Start:     3 5
# Default-Stop:       0 1 2 6
# Description:        Start STEP for test purposes
### END INIT INFO
```

В первой строке указывается имя программы или службы, которая контролируется данным скриптом. Во второй и третьей строках – имена служб, которые должны быть запущены или остановлены, прежде чем служба сама будет запущена или остановлена. В четвертой и пятой строках указаны уровни исполнения, на которых служба должна быть запущена или остановлена автоматически. Шестая строка – описание службы.

Для создания мягких ссылок в соответствующих каталогах уровней исполнения, необходимо пользоваться командой *insserv*. Например:

*insserv имя-скрипта*

Данная команда автоматически создает необходимые мягкие ссылки в каталогах соответствующих уровней исполнения. Не рекомендуется создавать такие ссылки вручную.

Для просмотра информации о том, какие службы и на каких уровнях исполнения включены или отключены, используется команда *chkconfig*. Например:

```
chkconfig --list
chkconfig имя-службы off
chkconfig имя-службы on
```

Первой командой мы просматриваем список всех служб и уровней исполнения, на которых они включены. Второй и третьей командой мы, соответственно, отключаем и включаем использование указанной службы в нашей системе.



Также полезно знать, что команды, указанные в файле `/etc/init.d/boot.local` будут автоматически выполняться системой перед переходом на какой-либо уровень исполнения. Работа данного файла аналогична работе файла `autoexec.bat` в операционной системе DOS

## 2.4 Задания для самопроверки

1. Запустите систему в однопользовательский режим работы с помощью GRUB.
2. С помощью загрузчика GRUB передайте команде `kernel` опцию `init=/bin/bash` и загрузите систему. Проанализируйте полученный результат.
3. Защитите паролем `P@ssw0rd` возможность редактирования пунктов меню загрузки GRUB без входа пользователя в систему.
4. Изучите список служб, которые запускаются в вашей системе на 5 уровне исполнения. Самостоятельно получите справку по этим службам и определитесь с тем, является ли их запуск для вас необходимым. Отключите запуск ненужных для вас служб
5. Настройте систему таким образом, чтобы в файле `/tmp/start.txt` хранились дата и время старта вашей системы. Информация в файле не должна накапливаться.
6. Настройте систему таким образом, чтобы в файле `/tmp/5Level.txt` хранились дата и время всех переходов вашей системы на 5 уровень исполнения. Информация в файле должна накапливаться.
7. В файле `/etc/inittab` присвойте параметру `initdefault` значение 6. Проанализируйте результат. Восстановите исходную настройку файла `/etc/inittab`.



### 3. Аппаратное обеспечение в Linux

Linux системы характерны тем, что развиваются достаточно быстрыми темпами. Тысячи людей, так или иначе, участвуют в разработке этой системы. В результате появляются новые идеи и их реализации, которые делают Linux лучше, стабильнее, удобнее и дружелюбнее.

#### 3.1 Каталог /dev

Сегодня в Linux приняты за основу два основных способа работы с аппаратным обеспечением. Первый способ основан на том, что для каждого потенциально возможного устройства заранее создается файл в каталоге `/dev`. Например, в классической схеме для жесткого диска, который подключен как Primary Master к IDE интерфейсу создается файл `/dev/hda`. Даже если в нашей системе нет такого устройства в данный момент, файл все равно будет создан.

Второй способ заключается в том, что для каталога `/dev` используется специализированная файловая система, которая генерирует только те файлы в данном каталоге, которые соответствуют реально существующим устройствам нашего компьютера. Это означает, что файл `/dev/hda` будет находиться в системе только в том случае, если действительно мы используем такой жесткий диск в данный момент. Следует сказать, данный метод является более современным способом организации каталога `/dev`.

Проанализируем вышесказанное. Командой `ls -a /dev` выведем на экран все содержимое каталога `/dev`. Среди файлов данного каталога нет файла `/dev/hda`. В таком случае логично предположить, что мы столкнулись с тем вариантом организации содержимого каталога `/dev`, в котором используется специализированная файловая система. Если это действительно так, то данная файловая система должна быть смонтирована непосредственно в каталог `/dev`. Командой `mount` можно подтвердить свои догадки. Среди полученного списка смонтированных файловых систем для нас интерес в данном случае представляет строка, которая выглядит так:

```
udev on /dev type tmpfs (rw)
```

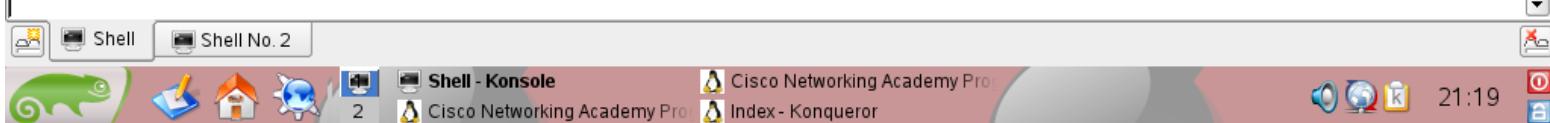
В этой строке указано, что некое устройство `udev`, которое использует файловую систему `tmpfs`, смонтировано в каталог `/dev` в режиме чтения и записи (`rw`).



```
Shell - Konsole
Session Edit View Bookmarks Settings Help

linux-step:/etc/syslog-# ls -a /dev
fd0u360 md13 net ptyq7 ptys5 sda tty24 tty51 ttyp5 ttyr3 usbdev1.1_ep00
fd0u720 md14 null ptyq8 ptys6 sda1 tty25 tty52 ttyp6 ttyr4 usbdev1.1_ep81
fd0u800 md15 port ptyq9 ptys7 sda2 tty26 tty53 ttyp7 ttyr5 vcs
fd0u820 md16 pop ptyqa ptys8 sda3 tty27 tty54 ttyp8 ttyr6 vcs1
fd0u830 md17 psaux ptyqb ptys9 sg0 tty28 tty55 ttyp9 ttyr7 vcs10
full md18 ptmx ptyqc ptysa shm tty29 tty56 ttypa ttyr8 vcs2
fwmonitor md19 pts ptyqd ptysb skip snapshot tty30 tty57 ttypb ttyr9 vcs3
hdc md2 ptyp0 ptyqe ptysc snd tty31 tty58 ttypc ttyra vcs4
hpet md20 ptyp1 ptyqf ptysd stderr tty32 tty59 ttypd ttyrb vcs5
i2c-0 md21 ptyp2 ptyr0 ptysf stdin tty33 tty60 ttypf ttyrd vcs7
initctl md22 ptyp3 ptyr1 ptysf stdout tty34 tty61 ttყ0 ttrye vcsa
input md23 ptyp4 ptyr2 ram0 tty35 tty62 ttყ1 ttryf vcsa1
kmem md24 ptyp5 ptyr3 ram1 tty36 tty63 ttყ2 ttყ0 vcsa10
kmsg md25 ptyp6 ptyr4 ram10 tty0 tty37 tty7 ttყ3 ttყ1 vcsa2
log md26 ptyp7 ptyr5 ram11 tty1 tty38 tty8 ttყ4 ttყ2 vcsa3
loop0 md27 ptyp8 ptyr6 ram12 tty10 tty39 tty9 ttყ5 ttყ3 vcsa4
loop1 md28 ptyp9 ptyr7 ram13 tty11 tty40 ttyS0 ttყ6 ttყ4 vcsa5
loop2 md29 ptypa ptyr8 ram14 tty12 tty41 ttyS1 ttყ7 ttყ5 vcsa6
loop3 md3 ptypb ptyr9 ram15 tty13 tty42 ttyS2 ttყ8 ttყ6 vcsa7
loop4 md30 ptypc ptyra ram2 tty14 tty43 ttyS3 ttყ9 ttყ7 watchdog
loop5 md31 ptypd ptyrb ram3 tty15 tty44 ttyS4 ttყa ttყ8 xconsole
loop6 md4 ptypf ptyrc ram4 tty16 tty45 ttyS5 ttყb ttყ9 zero
loop7 md5 ptypf ptyrd ram5 tty17 tty46 ttyS6 ttყc ttყa
lp0 md6 ptyq0 ptyre ram6 tty18 tty47 ttyS7 ttყd ttყb
mapper md7 ptyq1 ptyrf ram7 tty19 tty48 ttყe ttყc
md0 md8 ptyq2 ptys0 ram8 tty20 ttყf ttყd
md1 md9 ptyq3 ptys1 ram9 ttყ21 ttყ2 ttყe
md10 mem ptys4 ptys2 random ttყ22 ttყ3 ttყe
md11 midi ptys5 ptys3 route ttყ23 ttყ4 ttყ1
md12 mixer ptys6 ptys4 rtc ttყ5 ttყ3 ttყ1
ttყ6 ttყ4 ttყ2 urandom

linux-step:/etc/syslog-# mount
/dev/sda2 on / type reiserfs (rw,acl,user_xattr)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
debugfs on /sys/kernel/debug type debugfs (rw)
udev on /dev type tmpfs (rw)
devpts on /dev/pts type devpts (rw,mode=0620,gid=5)
/dev/sda3 on /home type ext3 (rw,acl,user_xattr)
securityfs on /sys/kernel/security type securityfs (rw)
/dev/hdc on /media/SU1020.001 type iso9660 (ro,nosuid,nodev,noatime,uid=0,utf8)
linux-step:/etc/syslog-#
```



Еще один плюс *tmpfs* – ее скорость. Поскольку файловая система *tmpfs* постоянно загружена в оперативную память, операции записи-чтения происходят почти мгновенно. Даже если интенсивно используется *swap*, скорость остается исключительно высокой. Более того, перемещение в *swap* означает передачу ресурсов процессам, наиболее нуждающимся в памяти, что способствует повышению общей производительности. Таким образом, свойство файловой системы *tmpfs* динамически изменять размер и, при необходимости, сбрасываться в *swap*, дает возможность операционной системе более гибко распоряжаться ресурсами.

Название файловой системы “*tmpfs*” свидетельствует об еще одном свойстве - как можно догадаться, данные в *tmpfs* после перезагрузки будут потеряны. Это свойство





можно считать как плюсом, так и минусом. Но независимо от точек мнения по этому вопросу, данный факт остается фактом.

Обратите внимание, несмотря на то, что устройство `/dev/hda` отсутствует, в нашей системе имеется устройство `/dev/hdc`. Команда `mount` проинформировала нас о том, что файловая система данного устройства смонтирована в каталог `/media/SU1020.001`, а тип файловой системы – `iso9660`. Очевидно, что мы имеем дело с компакт-диском, а `/dev/hdc` – DVD или CD привод, который подключен как Secondary Master к IDE интерфейсу.

Таким образом, из строки `udev on /dev type tmpfs (rw)` для нас нераскрыты осталась только опция “`udev`”. Но для того, чтобы получить полную картину того, что представляет собой данная опция, необходимо обратится к истории того, как эволюционировали способы организации каталога `/dev`.

### 3.2 Файлы устройств

Итак, `/dev` – это каталог, в котором хранятся файлы устройств. С помощью таких файлов ядро Linux создает интерфейс между пользовательскими приложениями и аппаратным обеспечением. Файлы устройств подразделяются на две группы, называемые *character devices* (символьные устройства) и *block devices* (блочные устройства). Первая группа содержит устройства, для которых отсутствует буферизация чтения/записи, а вторая группа, соответственно, содержит устройства, для которых чтение/запись буферизируется. Несмотря на названия, из обоих типов устройств может быть прочитан за раз один символ или блок. Поэтому такой способ присваивания имён может сбивать с толку и на самом деле неправилен.

Каждому устройству в Linux соответствуют два номера – верхний и нижний номер устройства (*minor* и *major*). Диапазоны доступных значений для *major* и *minor* номеров находятся в пределах 1-255. Именно эта пара номеров, а не имя устройств, является уникальным идентификатором устройства непосредственно для ядра. Для того чтобы возникало как можно меньше конфликтов и разнообразных коллизий, присвоение имен и номеров было проведено организацией LANANA (The Linux Assigned Names And Numbers Authority). Список всех устройств находится на сайте <http://www.lanaa.org>. Этот же список находится в файле `/usr/src/Linux/Documentation/devices.txt` вашей локальной системы, при условии, что на ней проинсталлированы исходные коды ядра:

- Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава
- Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков





Shell - Konsole

Session Edit View Bookmarks Settings Help

```
linux-step:/usr/src/linux/Documentation # head -40 devices.txt
    LINUX ALLOCATED DEVICES (2.6+ version)
    Maintained by Torben Mathiasen <device@lanana.org>
    Last revised: 15 May 2006

This list is the Linux Device List, the official registry of allocated
device numbers and /dev directory nodes for the Linux operating
system.

The latest version of this list is available from
http://www.lanana.org/docs/device-list/ or
ftp://ftp.kernel.org/pub/linux/docs/device-list/. This version may be
newer than the one distributed with the Linux kernel.

The LaTeX version of this document is no longer maintained.

This document is included by reference into the Filesystem Hierarchy
Standard (FHS). The FHS is available from http://www.pathname.com/fhs/.

Allocations marked (68k/Amiga) apply to Linux/68k on the Amiga
platform only. Allocations marked (68k/Atari) apply to Linux/68k on
the Atari platform only.

The symbol {2.6} means the allocation is obsolete and scheduled for
removal once kernel version 2.6 (or equivalent) is released. Some of these
allocations have already been removed.

This document is in the public domain. The author requests, however,
that semantically altered versions are not distributed without
permission of the author, assuming the author can be contacted without
an unreasonable effort.

In particular, please don't send patches for this list to Linus, at
least not without contacting me first.

I do not have any information about these devices beyond what appears
on this list. Any such information requests will be deleted without
reply.
```

linux-step:/usr/src/linux/Documentation #



Как видно из первых сорока строк файла *devices.txt*, данный документ курирует Torben Mathiasen, а все обновления для данного списка необходимо высыпать именно ему, а не, скажем, Линусу Торвальдсу, о чем и просит нас автор данного документа. В этом же файле указано местоположение более свежих версий данного документа. Обратите внимание, что данная версия документа справедлива для тех систем, ядро которых не младше версии 2.6.

Проанализируем, какие номера соответствуют нашему CD приводу */dev/hdc*, а также нашему SCSI диску */dev/sda*:

- Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава
- Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков





Shell - Konsole

Session Edit View Bookmarks Settings Help

```
linux-step:/usr/src/linux/Documentation # ls -l /dev/hdc
brw-r---- 1 root disk 22, 0 Jun 9 2008 /dev/hdc
linux-step:/usr/src/linux/Documentation # cat devices.txt | grep -m 25 "CD-ROM"
 3 block      First MFM, RLL and IDE hard disk/CD-ROM interface
               0 = /dev/hda          Master: whole disk (or CD-ROM)
               64 = /dev/hdb         Slave: whole disk (or CD-ROM)
11 block      SCSI CD-ROM devices
               0 = /dev/scd0        First SCSI CD-ROM
               1 = /dev/scd1        Second SCSI CD-ROM
12 block      MSCDEX CD-ROM callback support {2.6}
               0 = /dev/dos_cd0     First MSCDEX CD-ROM
               1 = /dev/dos_cd1     Second MSCDEX CD-ROM
15 block      Sony CDU-31A/CDU-33A CD-ROM
               0 = /dev/sonycd      Sony CDU-31a CD-ROM
16 block      GoldStar CD-ROM
               0 = /dev/gscd        GoldStar CD-ROM
17 block      Optics Storage CD-ROM
               0 = /dev/optcd       Optics Storage CD-ROM
18 block      Sanyo CD-ROM
               0 = /dev/sjcd        Sanyo CD-ROM
20 block      Hitachi CD-ROM (under development)
               0 = /dev/hitcd       Hitachi CD-ROM
22 block      Second IDE hard disk/CD-ROM interface
               0 = /dev/hdc          Master: whole disk (or CD-ROM)
               64 = /dev/hdd         Slave: whole disk (or CD-ROM)
23 block      Mitsumi proprietary CD-ROM
               0 = /dev/mcd         Mitsumi CD-ROM
24 block      Sony CDU-535 CD-ROM
linux-step:/usr/src/linux/Documentation # ls -l /dev/sda
brw-r---- 1 root disk 8, 0 Jun 9 2008 /dev/sda
linux-step:/usr/src/linux/Documentation # cat devices.txt | grep -m 10 "SCSI"
 8 block      SCSI disk devices (0-15)
               0 = /dev/sda         First SCSI disk whole disk
               16 = /dev/sdb        Second SCSI disk whole disk
               32 = /dev/sdc        Third SCSI disk whole disk
               240 = /dev/sdp       Sixteenth SCSI disk whole disk
 9 char       SCSI tape devices
               0 = /dev/st0         First SCSI tape, mode 0
               1 = /dev/st1         Second SCSI tape, mode 0
               32 = /dev/st01       First SCSI tape, mode 1
               33 = /dev/st11       Second SCSI tape, mode 1
linux-step:/usr/src/linux/Documentation # █
```

Shell Shell No. 2 Cisco Networking Academy Pro http://127.0.0.1 - Cisco Network 14:43

Из листинга видно, что устройство `/dev/hdc` является блочным (*block*), а не символьным (об этом свидетельствует символ “`b`”, который находится в самом начале строки вывода команды `ls -l /dev/hdc`). Цифры 22,0 той же строки указывают *major* и *minor* номера данного устройства. Просмотрев файл `devices.txt` мы убедились в том, что 22-й *major* номер блочного устройства соответствует именно второму IDE интерфейсу. Как мы знаем, к такому интерфейсу могут быть подключены 2 устройства: *minor* номер 0 определяет Master, а *minor* номер 64 – Slave. Наш SCSI диск также является блочным устройством с номерами 8,0. Это, согласно файлу `devices.txt`, соответствует первому SCSI диску системы. В листинге также представлен пример символьного устройства (*char*) – SCSI tape.



Итак, метод статического назначения имен для устройств сопряжен с рядом трудностей:

- В каталоге */dev* находится неимоверно большое количество разнообразных файлов, и их число постоянно растет с появлением поддержки новых устройств.
- Количество новых устройств постоянно растет и неизбежна ситуация, связанная с дефицитом диапазонов адресов.

Для решения этих проблем в ядре Linux версии 2.4 была внедрена файловая система *devfs*. Она предоставляет доступ пользователям к устройствам, добавляя новые файлы устройств по мере их обнаружения. Однако решив вышеперечисленные проблемы, *devfs* породила новые, из-за чего разработчики ядра прекратили ее поддержку. Для этого было несколько причин:

- **Громоздкая и неудобная система наименования устройств.**

На замену привычным и удобным конструкциям типа */dev/hda* пришли сложные конструкции типа */dev/ide/host0/bus0/target0/Lun0/disc*. Кроме того, такие «новые» имена не соответствуют именам, которые определены в LANANA. Что требовало реализации специального демона *devfsd*, который исполнял роль дополнительного слоя совместимости.

- **Невозможность динамического распределения номеров устройств.**

На практике в *devfs* используются номера, которые определены в LANANA. И хотя возможность реализации динамической адресации в *devfs* теоретически была, стабильные практические реализации не были сделаны. Таким образом, вопрос о дефиците диапазонов номеров остался открытым даже при внедрении *devfs* в Linux.

- **«Размножение» устройств при многократном подключении USB дисков.**

Несмотря на то, что каждый USB диск обнаруживается системой и настраивается для дальнейшей работы с ним, для каждого нового диска создается новый файл устройства, что влечет за собой невозможность автоматического монтирования нового диска аналогично предыдущему. Безусловно, ничто не мешает монтировать диски вручную, но именно это, возможно, «нервирует» пользователей Linux больше всего.

Помимо этих проблем, ядра версии 2.4 имели еще и проблемы в используемой драйверной модели. К этим проблемам относятся:

Киев	Днепропетровск	Львов	Ровно	Мариуполь	Полтава
Одесса	Донецк	Николаев	Запорожье	Луганск	Харьков



- Отсутствие единого метода представления связей между модулями ядра (драйверами) и устройствами
- Отсутствие общего механизма горячего подключения для таких шин, как USB и Firewire.

В результате, в современных системах, построенных на базе Linux ядра версии 2.6, используется новая драйверная модель, что повлекло за собой и изменения в принципах формировании каталога `/dev`. А наиболее заметным нововведением является внедрение файловой системы `sysfs`.

### 3.3 SysFS

Начиная с ядер версии 2.6 все устройства в Linux доступны пользователям через файловую систему `sysfs`. Она реализована как часть ядра и не может быть представлена модулем. Совместно с `sysfs` работает еще три подсистемы:

- *namedev*

Работает с именами устройств. Обеспечивается совместимость при использовании в системе различных принципов именования устройств.

- *libsfs*

набор библиотек, которые можно использовать для доступа к информации, которая хранится в `sysfs`

- *udev*

Обеспечивает динамическую организацию каталога `/dev`. Каждый раз, когда устройство добавляется или удаляется, ядро посылает специальное уведомление `udev`. События по удалению или добавлению устройств можно отобразить командой `udevadm monitor`. Такую информацию `udev` может посыпать также службе `syslog`, которая журнализирует все события, происходящие в системе.

Для получения доступа к `sysfs` необходимо смонтировать ее к каталогу `/sys` командой `mount -t sysfs sysfs /sys`. В результате в каталоге `/sys` нами будет обнаружена структура, представленная в виде дерева согласно определенной иерархии.





Shell No. 2 - Konsole

Session Edit View Bookmarks Settings Help

```
linux-step:~ # ls -a /sys/*
/sys/block:
.  fd0  loop0  loop2  loop4  loop6  ram0  ram10  ram12  ram14  ram2  ram4  ram6  ram8  sda
..  hdc  loop1  loop3  loop5  loop7  ram1  ram11  ram13  ram15  ram3  ram5  ram7  ram9

/sys/bus:
.  ... ac97  acpi  gameport  i2c  ide  pci  pci_express  platform  pnp  scsi  serio  spi  usb

/sys/class:
.  graphics  mem  pci_bus      scsi_disk      sound      spi_transport  usb_endpoint  vtconsole
..  i2c-adapter  misc  printer    scsi_generic  spi_host     tty          usb_host
dma  input       net  scsi_device  scsi_host    spi_master   usb_device   vc

/sys/devices:
.  ... acpi  pci0000:00  platform  pnp0  system  virtual

/sys/firmware:
.  ... acpi  edd

/sys/fs:
.  ...

/sys/kernel:
.  ... debug  kexec_crash_loaded  kexec_loaded  security  uevent_helper  uevent_seqnum

/sys/module:
.  button      ide_cd      mbcache      pcnet32      shpchp      snd_seq
..  cdrom      ide_core    md_mod      piix        snd        snd_seq_device
8250  dm_mod      ide_disk    mii         printk      snd_ac97_bus  snd_seq_midi
aamatch_pcrc  edd       ide_generic  mousedev   processor  snd_ac97_codec  snd_seq_midi_event
ac      ext3       intel_agp   mptbase     psmouse    snd_ens1371  snd_timer
af_packet  fan       ipv6       mptscsih   reiserfs   snd_mixer_oss  soundcore
agpgart   gameport  jbd       mptspi     scsi_mod    snd_page_alloc  thermal
apm     i2c_core  keyboard   parport    scsi_transport_spi  snd_pcm    uhci_hcd
apparmor  i2c_piix4  loop      parport_pc  sd_mod     snd_pcm_oss   usbcore
battery   i8042     lp       pci_hotplug  sg        snd_rawmidi

/sys/power:
.  ... disk  image_size  resume  state
linux-step:~ # cat /sys/class/net/eth0/address
00:0c:29:fa:71:4a
linux-step:~ #
```

Shell

Shell No. 2

03:18

Итак, каждое устройство или драйвер в *sysfs* представлен определенным каталогом. Файлы этих каталогов являются атрибутами драйверов или устройств. На примере видно, что в файле */sys/class/net/eth0/address* хранится MAC-адрес Ethernet адаптера *eth0*. Для каждого атрибута создается свой собственный файл. Назначение подкаталогов в каталоге */sys* следующее:

- **/sys/block**

Содержит каталоги блочных устройств нашей системы. Для того, чтобы устройство было представлено здесь своим каталогом, необходим не только сам факт наличия устройства, но и наличие в оперативной памяти необходимых для его работы драйверов. Например, если к системе подключен USB-диск, то в каталоге





`/sys/devices` он появится в любом случае, а в каталоге `/sys/bus/blk` только при условии, что загружены необходимые для поддержки usb драйверы.

- **`/sys/bus`**

Содержит перечень шин, которые зарегистрированы в ядре. В каждом каталоге шины присутствуют каталоги `devices` и `drivers`:

- о `devices` – мягкие ссылки на каталоги устройств, к даннойшине.
- о `drivers` – каталоги драйверов, загруженных для устройств.

При обнаружении драйвером «своего» устройства в каталоге, появляются мягкие ссылки на каталог этого устройства и на модуль, который используется для его работы.

```
Shell No. 2 - Konsole
Session Edit View Bookmarks Settings Help
linux-step:~ # ls /sys/bus/
ac97 acpi gameport i2c ide pci pci_express platform pnp scsi serio spi usb
linux-step:~ # ls -l /sys/bus/pci/*
/sys/bus/pci/devices:
total 0
lrwxrwxrwx 1 root root 0 Jun 11 03:05 0000:00:00.0 -> ../../../../devices/pci0000:00/0000:00:00.0
lrwxrwxrwx 1 root root 0 Jun 11 04:14 0000:00:01.0 -> ../../../../devices/pci0000:00/0000:00:01.0
lrwxrwxrwx 1 root root 0 Jun 11 04:14 0000:00:07.0 -> ../../../../devices/pci0000:00/0000:00:07.0
lrwxrwxrwx 1 root root 0 Jun 11 04:14 0000:00:07.1 -> ../../../../devices/pci0000:00/0000:00:07.1
lrwxrwxrwx 1 root root 0 Jun 11 04:14 0000:00:07.2 -> ../../../../devices/pci0000:00/0000:00:07.2
lrwxrwxrwx 1 root root 0 Jun 11 04:14 0000:00:07.3 -> ../../../../devices/pci0000:00/0000:00:07.3
lrwxrwxrwx 1 root root 0 Jun 11 04:14 0000:00:0f.0 -> ../../../../devices/pci0000:00/0000:00:0f.0
lrwxrwxrwx 1 root root 0 Jun 11 04:14 0000:00:10.0 -> ../../../../devices/pci0000:00/0000:00:10.0
lrwxrwxrwx 1 root root 0 Jun 11 04:14 0000:00:11.0 -> ../../../../devices/pci0000:00/0000:00:11.0
lrwxrwxrwx 1 root root 0 Jun 11 04:14 0000:00:12.0 -> ../../../../devices/pci0000:00/0000:00:12.0

/sys/bus/pci/drivers:
total 0
drwxr-xr-x 2 root root 0 Jun 11 04:15 ENS1371
drwxr-xr-x 2 root root 0 Jun 11 04:15 PIIIX_IDE
drwxr-xr-x 2 root root 0 Jun 11 04:15 agpgart-intel
drwxr-xr-x 2 root root 0 Jun 11 04:15 imsttftb
drwxr-xr-x 2 root root 0 Jun 11 04:15 mptsp
drwxr-xr-x 2 root root 0 Jun 11 04:15 parport_pc
drwxr-xr-x 2 root root 0 Jun 11 04:15 pcieport-driver
drwxr-xr-x 2 root root 0 Jun 11 04:15 pcnet32
drwxr-xr-x 2 root root 0 Jun 11 04:15 piix4_smbus
drwxr-xr-x 2 root root 0 Jun 11 04:15 serial
drwxr-xr-x 2 root root 0 Jun 11 04:15 shpchp
drwxr-xr-x 2 root root 0 Jun 11 04:15 uhci_hcd

/sys/bus/pci/slots:
total 0
linux-step:~ # ls -l /sys/bus/pci/drivers/ENS1371
total 0
lrwxrwxrwx 1 root root 0 Jun 11 04:15 0000:00:12.0 -> ../../../../../../devices/pci0000:00/0000:00:12.0
--w----- 1 root root 4096 Jun 11 04:15 bind
lrwxrwxrwx 1 root root 0 Jun 11 04:15 module -> ../../../../../../module/snd_ens1371
--w----- 1 root root 4096 Jun 11 04:15 new_id
--w----- 1 root root 4096 Jun 11 04:15 unbind
linux-step:~ #
```



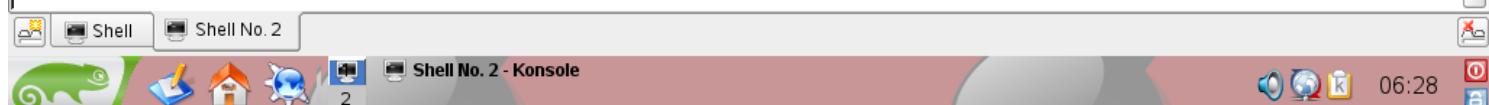


- **/sys/class**

Отражает устройства, сгруппировав их в соответствующие классы. Подразумевается наличие и устройства и его драйвера в системе. Так, например, в каталоге `/sys/class/net` представлены все сетевые интерфейсы нашей системы. По каждому сетевому интерфейсу можно получить обширную информацию, а также статистику. Например, для нашего Ethernet адаптера `eth0` статистика находится в каталоге `/sys/class/net/eth0/statistics`:

```
Shell No. 2 - Konsole
Session Edit View Bookmarks Settings Help

linux-step:~ # ls /sys/class/
dma           input   net      scsi_device  scsi_host  spi_master    usb_device   vc
graphics      mem     pci_bus  scsi_disk   sound      spi_transport  usb_endpoint  vtconsole
i2c-adapter   misc    printer  scsi_generic  spi_host   tty          usb_host
linux-step:~ # ls /sys/class/net
eth0  lo  sit0
linux-step:~ # ls -l /sys/class/net/eth0/
total 0
-r--r--r-- 1 root root 4096 Jun 11 03:15 addr_len
-r--r--r-- 1 root root 4096 Jun 11 03:15 address
-r--r--r-- 1 root root 4096 Jun 11 03:15 broadcast
-r--r--r-- 1 root root 4096 Jun 11 03:15 carrier
lrwxrwxrwx  1 root root    0 Jun 11 03:15 device -> ../../devices/pci0000:00/0000:00:11.0
-r--r--r-- 1 root root 4096 Jun 11 03:15 dormant
-r--r--r-- 1 root root 4096 Jun 11 03:15 features
-rw-r--r-- 1 root root 4096 Jun 11 03:15 flags
-r--r--r-- 1 root root 4096 Jun 11 03:15 ifindex
-r--r--r-- 1 root root 4096 Jun 11 03:15 iflink
-r--r--r-- 1 root root 4096 Jun 11 03:15 link_mode
-rw-r--r-- 1 root root 4096 Jun 11 03:15 mtu
-r--r--r-- 1 root root 4096 Jun 11 03:15 operstate
drwxr-xr-x  2 root root    0 Jun 11 04:39 statistics
lrwxrwxrwx  1 root root    0 Jun 11 03:15 subsystem -> ../../class/net
-rw-r--r-- 1 root root 4096 Jun 11 03:15 tx_queue_len
-r--r--r-- 1 root root 4096 Jun 11 03:15 type
--w----- 1 root root 4096 Jun 11 03:15 uevent
-rw-r--r-- 1 root root 4096 Jun 11 03:15 weight
linux-step:~ # ls /sys/class/net/eth0/statistics/
collisions      rx_crc_errors  rx_frame_errors  rx_packets      tx_compressed   tx_heartbeat_errors
multicast       rx_dropped    rx_length_errors  tx_aborted_errors tx_dropped      tx_packets
rx_bytes        rx_errors     rx_missed_errors  tx_bytes       tx_errors      tx_window_errors
rx_compressed   rx_fifo_errors rx_over_errors   tx_carrier_errors tx_fifo_errors
linux-step:~ # cat /sys/class/net/eth0/statistics/tx_packets
1542
linux-step:~ # cat /sys/class/net/eth0/statistics/rx_packets
441538
linux-step:~ # cat /sys/class/net/eth0/statistics/rx_bytes
41612849
linux-step:~ # cat /sys/class/net/eth0/statistics/tx_bytes
211748
linux-step:~ # ■
```



В приведенном примере мы получили информацию о том, сколько сетевых пакетов было отправлено и получено нашим сетевым адаптером (`tx_packets` и `rx_packets` соответственно), а также информацию о том, сколько им было получено и отправлено байт информации (`tx_bytes` и `rx_bytes`).



- **/sys/devices**

Содержит дерево устройств, которое отражает связь между устройствами и соответствует внутреннему дереву устройств ядра. Мягкие ссылки в подкаталогах могут указывать шину устройства, его класс и загруженный драйвер.

- **/sys/module**

В данном каталоге можно узнать и, иногда, изменить атрибуты загруженных модулей.

- **/sys/power**

Используется, в частности, для suspend режима системы.

Реализация драйверной модели в Linux претерпевает постоянные изменения. Сегодня сложно сказать, насколько долговременной будет система, которая является стандартом Linux ядер версии 2.6. Однако можно с уверенностью сказать, что разработчики Linux постоянно ищут новые решения вопросов взаимодействия пользователя с аппаратным обеспечением. Очевидно, что сегодняшняя реализация, основанная на файловой системе *sysfs*, лучше предыдущей, основанной на файловой системе *devfs*. Поэтому, если есть основания ожидать, что в ядрах Linux версии 2.8 будут реализованы новые механизмы, они будут еще лучше и современнее тех, что реализованы сегодня.

### 3.4 Задания для самопроверки

1. Проанализируйте каталог */dev* вашей системы. Какое количество файлов для разделов жесткого диска там создано? Сопоставьте эту информацию с выводом команды *fdisk -L*.
2. Изучите содержимое каталога */sys* вашей системы. Определите MAC-адрес сетевых адаптеров, установленных в вашей системе
3. Подключите USB-флешку к вашему компьютеру и проанализируйте события, которые происходят в вашей системе для того, чтобы это устройство было доступно для работы.





## 4. Управление пользователями

Система Linux похожа на монархическое государство, в котором существует один суперпользователь - `root`, которому все подчиняется, и определенное число обычных пользователей. Это значит, что если вы попробуете удалить один из жизненно важных файлов Linux под учетной записью пользователя, то система не позволит вам этого сделать. Структура файловой системы Linux такова, что она является более приспособленной к неподготовленному или неграмотному в своих действиях пользователю, работа такого пользователя теоретически может принести гораздо меньшее число бед, чем в ОС Windows.

Под учетной записью `root` вы имеете право делать все, что вам заблагорассудится. Поэтому в целях безопасности, даже если вы используете систему в гордом одиночестве, настоятельно рекомендуется создавать в системе хотя бы одного пользователя. И даже вам следует работать под пользователем, а не суперпользователем, дабы что-нибудь случайно не испортить.

Правильное управление учетными записями является залогом безопасности системы. Редко используемые учетные записи становятся главными мишенями атак хакеров, как и те записи, пароли к которым легко подобрать. Даже если для подключения и удаления пользователей применяются автоматизированные системные утилиты, важно понимание того, какие проходят при этом изменения в системе.

Информация о пользователях и группах хранится в нескольких файлах, которые и будут нами рассмотрены в следующих разделах данной главы.

### 4.1 /etc/passwd

Данный файл содержит список пользователей, которые известны системе. В процессе регистрации пользователя, система обращается к этому файлу в поисках идентификатора пользователя, а также с целью проверки введенного пароля. Каждая строка файла описывает одного пользователя и содержит семь полей, разделенных двоеточиями:

- Киев
- Днепропетровск
- Львов
- Ровно
- Мариуполь
- Полтава
- Одесса
- Донецк
- Николаев
- Запорожье
- Луганск
- Харьков





Shell - Konsole

Session Edit View Bookmarks Settings Help

```
SuSe:~ # cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:Daemon:/sbin:/bin/bash
lp:x:4:7:Printing daemon:/var/spool/lpd:/bin/bash
mail:x:8:12:Mailer daemon:/var/spool/clientmqueue:/bin/false
news:x:9:13:News system:/etc/news:/bin/bash
uucp:x:10:14:Unix-to-Unix CoPy system:/etc/uucp:/bin/bash
games:x:12:100:Games account:/var/games:/bin/bash
man:x:13:62:Manual pages viewer:/var/cache/man:/bin/bash
at:x:25:25:Batch jobs daemon:/var/spool/at/jobs:/bin/bash
wwwrun:x:30:8:WWW daemon apache:/var/lib/wwwrun:/bin/false
squid:x:31:65534:WWW-proxy squid:/var/cache/squid:/bin/false
irc:x:39:65534:IRC daemon:/usr/lib/ircd:/bin/bash
ftp:x:40:49:FTP account:/srv/ftp:/bin/bash
named:x:44:44:Name server daemon:/var/lib/named:/bin/false
gdm:x:50:15:Gnome Display Manager daemon:/var/lib/gdm:/bin/bash
postfix:x:51:51:Postfix Daemon:/var/spool/postfix:/bin/false
mysql:x:60:2:MySQL database admin:/var/lib/mysql:/bin/false
pop:x:67:100:POP admin:/var/lib/pop:/bin/false
sshd:x:71:65:SSH daemon:/var/lib/sshd:/bin/false
mailman:x:72:67:GNU mailing list manager:/var/lib/mailman:/bin/bash
ntp:x:74:65534:NTP daemon:/var/lib/ntp:/bin/false
ldap:x:76:70:User for OpenLDAP:/var/lib/ldap:/bin/bash
radiusd:x:100:101:Radius daemon:/var/lib/radiusd:/bin/false
privoxy:x:101:102:Daemon user for privoxy:/var/lib/privoxy:/bin/false
vdr:x:102:33:Video Disk Recorder:/var/spool/video:/bin/false
nobody:x:65534:65533:nobody:/var/lib/nobody:/bin/bash
oleg:x:500:100:oleg:/home/oleg:/bin/bash
alla:x:501:100:Alla:/home/alla:/bin/bash
SuSe:~ #
```

Каждое из семи полей файла */etc/passwd* имеет свое строгое предназначение:

*username : password : UserID : GroupID : fullname : homedir : shell*

## 1. Имя пользователя (username)

Регистрационное имя пользователя, то есть логин.

## 2. Пароль (password)

В этом поле должен отображаться пароль в незашифрованном или зашифрованном виде. Ввиду неактуальности такого вида хранения паролей,

- |          |                  |            |             |             |           |
|----------|------------------|------------|-------------|-------------|-----------|
| ■ Киев   | ■ Днепропетровск | ■ Львов    | ■ Ровно     | ■ Мариуполь | ■ Полтава |
| ■ Одесса | ■ Донецк         | ■ Николаев | ■ Запорожье | ■ Луганск   | ■ Харьков |



в современных дистрибутивах в данном поле пароль не указывается. Для этого была создана схема "теневых паролей", основанная на */etc/shadow*.

### 3. Идентификатор пользователя (User ID)

Индивидуальный числовой идентификатор пользователя (UID). Система обычно работает с UID, а не с именем пользователя. Идентификатор задается из диапазона 0...65534 и должен быть уникальным. Число 0 соответствует пользователю root. Желательно идентификаторы назначать не произвольным образом, а системно. Например, выделить определенный интервал (1000...1100) под одну группу пользователей, а еще один (2000...2100) - под другую группу. В каждом диапазоне назначать идентификаторы следует последовательно. Это опять же упростит администрирование и позволит, бегло взглянув на список процессов, сразу определить кто чем занимается.

### 4. Идентификатор группы (GroupID)

Числовой идентификатор первичной группы пользователя (GID). Помимо первичной группы, пользователь может входить или не входить в состав разных групп, но в первичную группу (native group) он всегда входит. В различных дистрибутивах это выглядит по-разному. Например, в RedHat дистрибутивах для каждого пользователя при его создании создается автоматически приватная группа, которая имеет такое же имя, как и имя пользователя. В данную группу по умолчанию входит только один пользователь. В SuSE дистрибутивах пользователи входят в группу users, что позволяет сократить количество групп, неизбежно увеличивающих свое количество, при создании приватной группы для каждого пользователя. Идентификатор группы 0 соответствует группе root

### 5. Реальное имя пользователя (full name)

Обычно представляет собой фактическое имя, например Linus Torvalds. Может содержать и другие данные, номер телефона и т.д. Эти сведения используются исключительно в информационных целях.

### 6. Домашний каталог пользователя (home dir)

В качестве домашнего каталога обычно используется каталог */home/имя\_пользователя*. Без особых причин не следует изменять такую организацию каталогов (ваш преемник на должности администратора будет вам за это очень признателен)

- Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава
- Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков



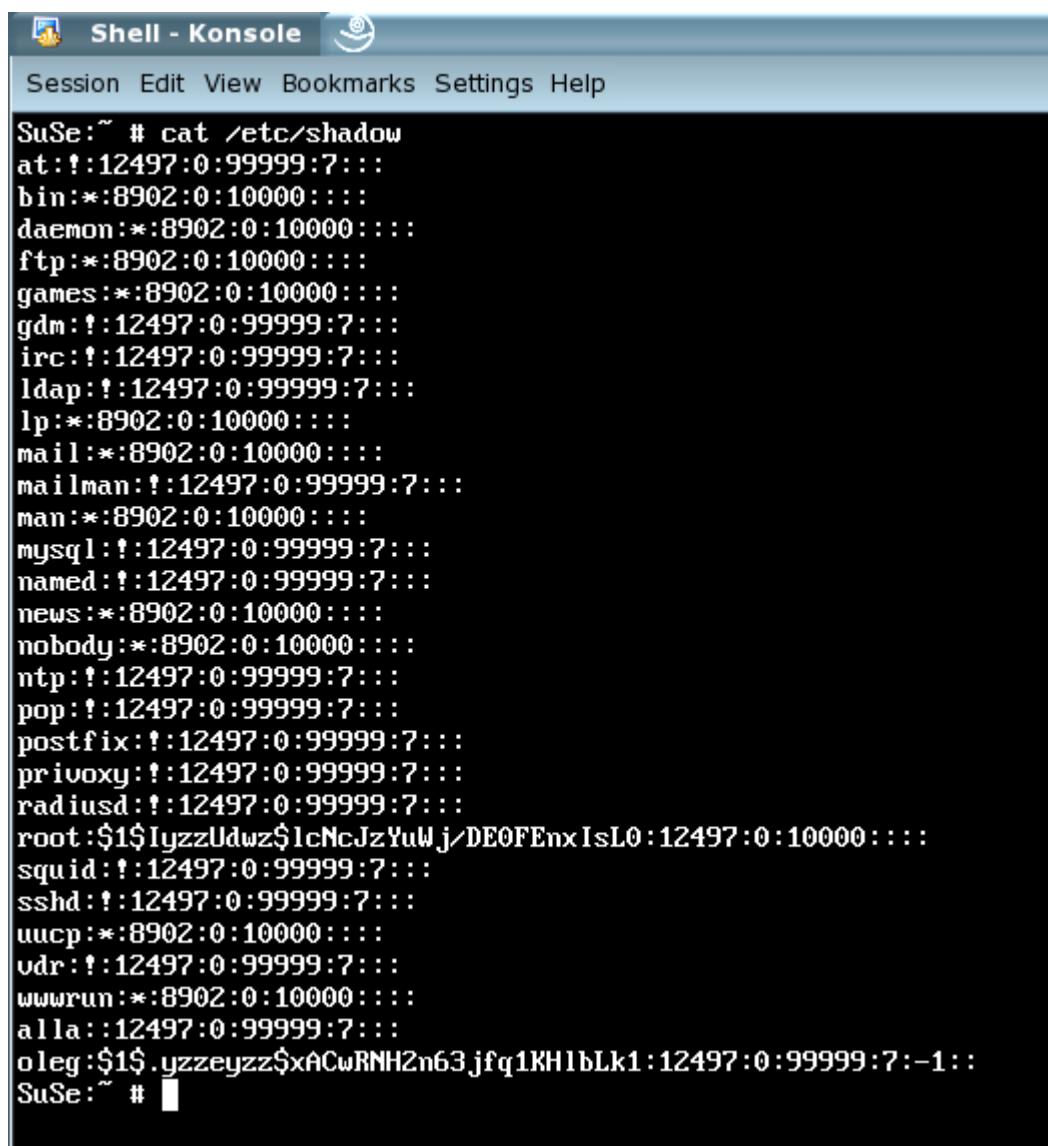


## 7. Оболочка пользователя (login shell)

Командный интерпретатор пользователя, который используется им по умолчанию. Программа-оболочка (командный интерпретатор) запускается при входе пользователя в систему. Примеры командных интерпретаторов: ash, bash, csh, ksh

### 4.2 /etc/shadow

В отличии от файла */etc/passwd*, который должен быть открыт для чтения, а значит и для копирования, к файлу */etc/shadow* открыт доступ только суперпользователю. Кроме того, дополнительные поля данного файла, позволяют осуществить тонкую настройку временных параметров учетной записи.



```
SuSe:~ # cat /etc/shadow
at:!:12497:0:99999:7:::
bin:*:8902:0:10000::::
daemon:*:8902:0:10000::::
ftp:*:8902:0:10000::::
games:*:8902:0:10000::::
gdm:!:12497:0:99999:7:::
irc:!:12497:0:99999:7:::
ldap:!:12497:0:99999:7:::
lp:*:8902:0:10000::::
mail:*:8902:0:10000::::
mailman:!:12497:0:99999:7:::
man:*:8902:0:10000::::
mysql:!:12497:0:99999:7:::
named:!:12497:0:99999:7:::
news:*:8902:0:10000::::
nobody:*:8902:0:10000::::
ntp:!:12497:0:99999:7:::
pop:!:12497:0:99999:7:::
postfix:!:12497:0:99999:7:::
privoxy:!:12497:0:99999:7:::
radiusd:!:12497:0:99999:7:::
root:$1$IyzzUdwz$1cNcJzYuWj/DE0FEnxIsL0:12497:0:10000::::
squid:!:12497:0:99999:7:::
sshd:!:12497:0:99999:7:::
uucp:*:8902:0:10000::::
vdr:!:12497:0:99999:7:::
wwwrun:*:8902:0:10000::::
alla:!:12497:0:99999:7:::
oleg:$1$.yzseyzz$xCwRNH2n63jfq1KH1bLk1:12497:0:99999:7:-1:::
SuSe:~ #
```





Рассмотрим назначение полей данного файла:

*username: password: date: min: Max: Warning: Inactive: Expired: reserved*

**1. Регистрационное имя пользователя (username)**

Должно в точности соответствовать имени пользователя в файле */etc/passwd*.

**2. Хешированный пароль длиной от 13 до 34 ASCII символов (password)**

Допустимые символы - a-z, A-Z, 0-9, '.', '\$ и '/'.

**3. Дата последнего изменения пароля (date)**

Число дней, прошедших с начала эпохи Unix (1 января 1970 года - запомните эту дату!)

**4. Минимальный срок жизни пароля (min)**

Число дней, которое должно пройти с момента последнего изменения пароля, прежде чем его снова можно будет поменять

**5. Максимальный срок жизни пароля (Max)**

Число дней, которое должно пройти с момента последнего изменения пароля, когда пользователь обязан будет сменить пароль

**6. Предупреждение (Warning)**

Число дней до истечения срока действия пароля, когда выдается предупреждение

**7. Блокировка (Inactive)**

Число дней по истечению срока действия пароля, когда учетная запись блокируется.

**8. Срок жизни учетной записи (Expired)**

Число дней, которое должно пройти с начала эпохи Unix, чтобы учетная запись устарела. В полночь последнего дня учетная запись будет заблокирована

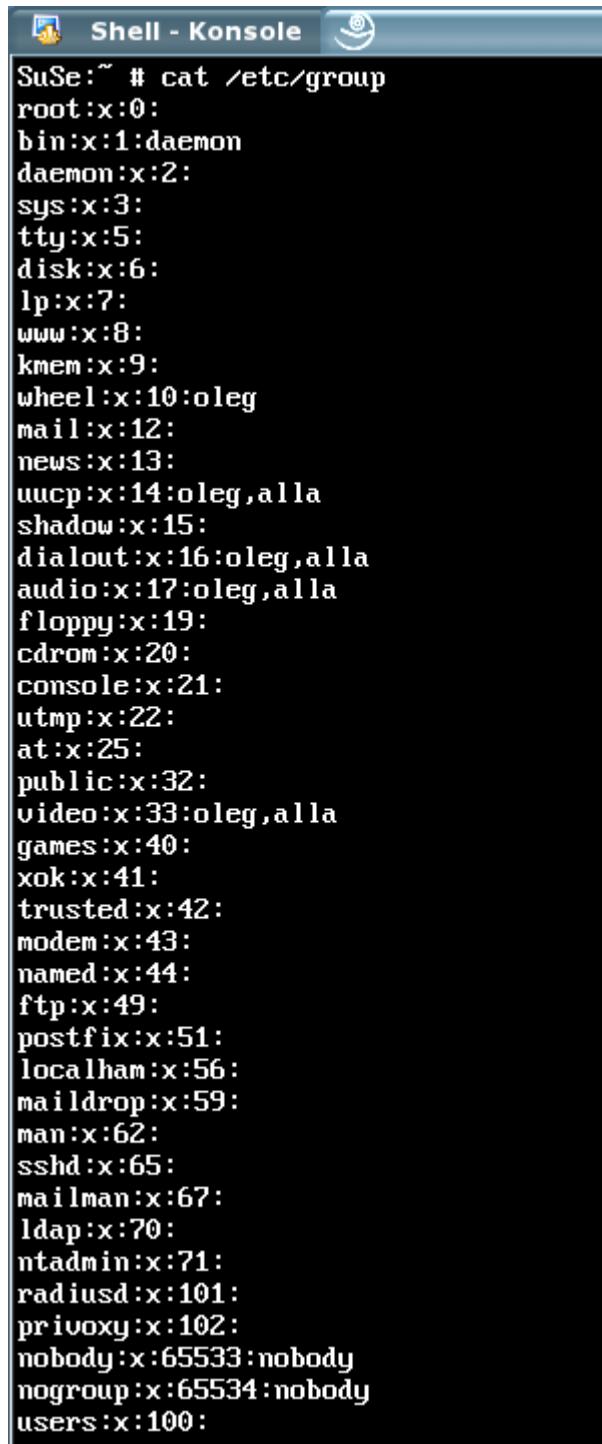
**9. Зарезервировано (reserved)**

Возможно, это поле в будущем будет как-то использоваться.



## 4.3 /etc/group

Файл групп - файл, в котором перечислены все зарегистрированные группы, а так же пользователи, которые входят в заданные группы:



```
SuSe:~ # cat /etc/group
root:x:0:
bin:x:1:daemon
daemon:x:2:
sys:x:3:
tty:x:5:
disk:x:6:
lp:x:7:
www:x:8:
kmem:x:9:
wheel:x:10:oleg
mail:x:12:
news:x:13:
uucp:x:14:oleg,alla
shadow:x:15:
dialout:x:16:oleg,alla
audio:x:17:oleg,alla
floppy:x:19:
cdrom:x:20:
console:x:21:
utmp:x:22:
at:x:25:
public:x:32:
video:x:33:oleg,alla
games:x:40:
xok:x:41:
trusted:x:42:
modem:x:43:
named:x:44:
ftp:x:49:
postfix:x:51:
localham:x:56:
maildrop:x:59:
man:x:62:
sshd:x:65:
mailman:x:67:
ldap:x:70:
ntadmin:x:71:
radiusd:x:101:
privoxy:x:102:
nobody:x:65533:nobody
nogroup:x:65534:nobody
users:x:100:
```



Назначение полей:

*groupname : password : GID : users*

#### **1. Имя группы (groupname)**

Напомню, что зачастую имя группы совпадает с именем зарегистрированного пользователя, который и входит в данную группу.

#### **2. Пароль группы (password)**

Хранится в современных дистрибутивах не в данном файле по той же причине что и пароли пользователей не в файле */etc/passwd*.

#### **3. Идентификатор группы (GID)**

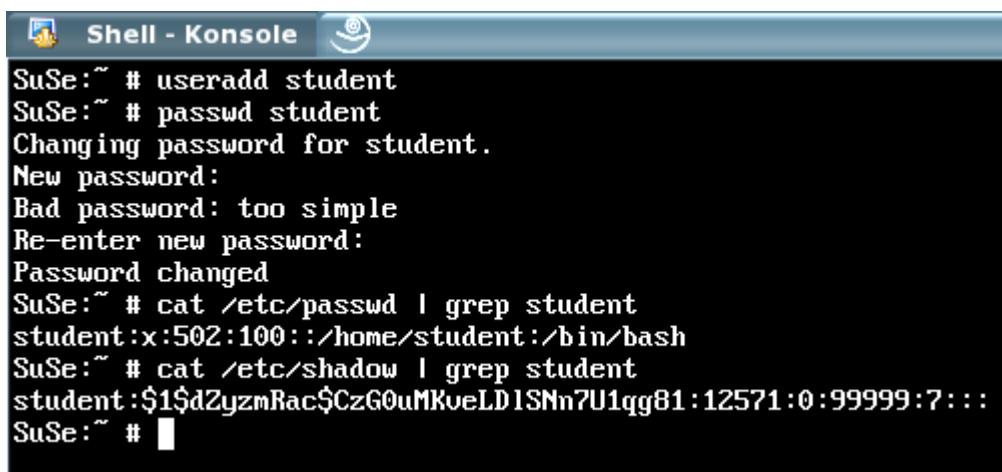
Именно данный идентификатор указывается в соответствующем поле файла паролей

#### **4. Пользователи группы (users)**

Если в группе больше одного пользователя, то в данном поле будет проводиться их перечисление через запятую.

## **4.4 Управление пользователями**

Для создания учетной записи используется команда *adduser имя\_пользователя* или *useradd имя\_пользователя*.



```
Shell - Konsole
SuSe:~ # useradd student
SuSe:~ # passwd student
Changing password for student.
New password:
Bad password: too simple
Re-enter new password:
Password changed
SuSe:~ # cat /etc/passwd | grep student
student:x:502:100::/home/student:/bin/bash
SuSe:~ # cat /etc/shadow | grep student
student:$1$ZyzmRac$CzG0uMKveLD1SNn7U1qg81:12571:0:99999:7:::
SuSe:~ #
```

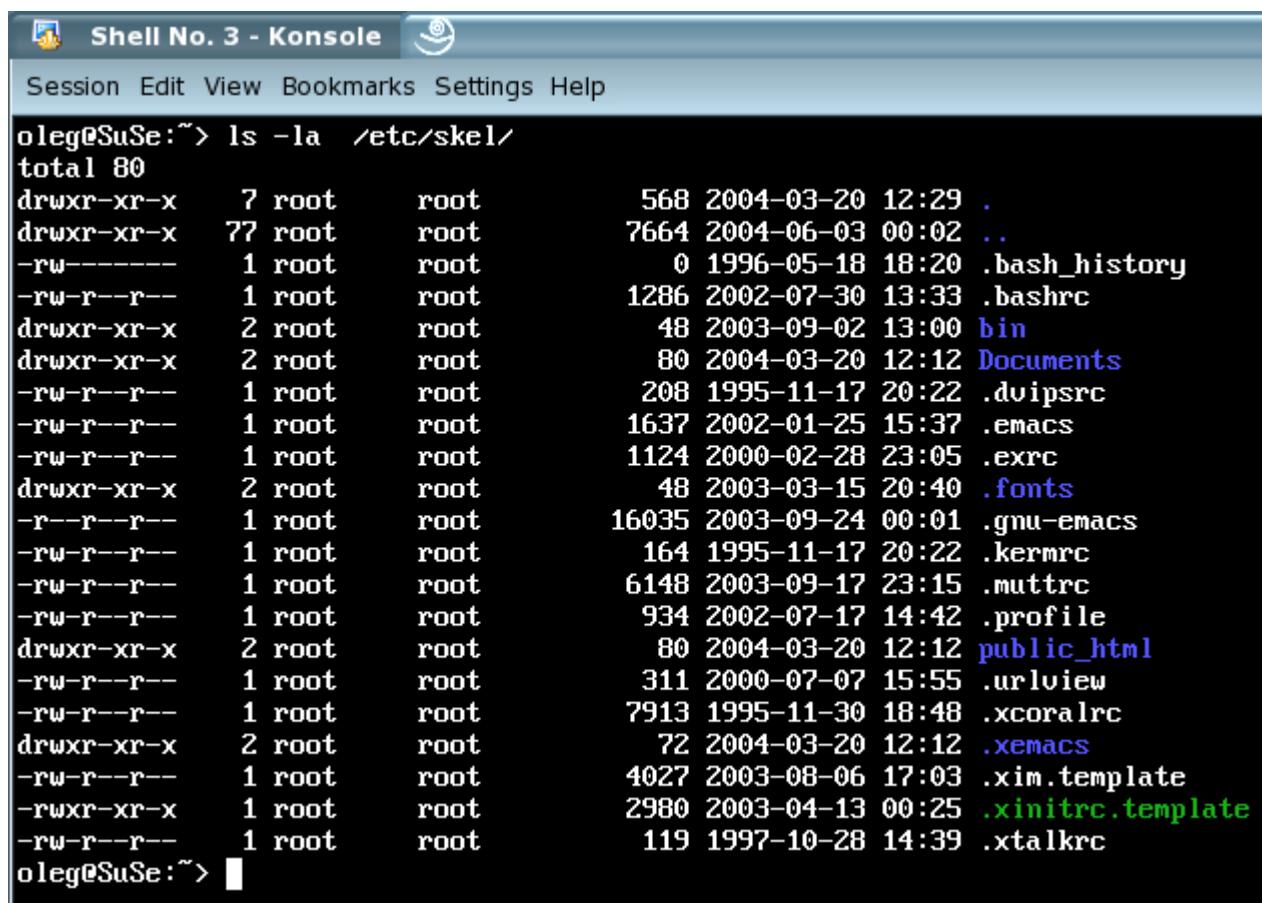
- Киев
- Днепропетровск
- Львов
- Ровно
- Мариуполь
- Полтава
- Одесса
- Донецк
- Николаев
- Запорожье
- Луганск
- Харьков





После создания учетной записи не забудьте изменить пароль пользователя командой `passwd имя_пользователя`. При выборе нового пароля система может выдавать вам различные сообщения о простоте вводимого вами пароля или его небольшой длине. Однако если вы операции по смене пароля производите под суперпользователем - вы можете просто проигнорировать данные сообщения. При использовании команды `passwd` пароль заносится в файл скрытых паролей.

При создании пользователя, для него, как правило, автоматически создается домашний каталог, содержимое которого полностью повторяет содержимое каталога `/etc/skel`:

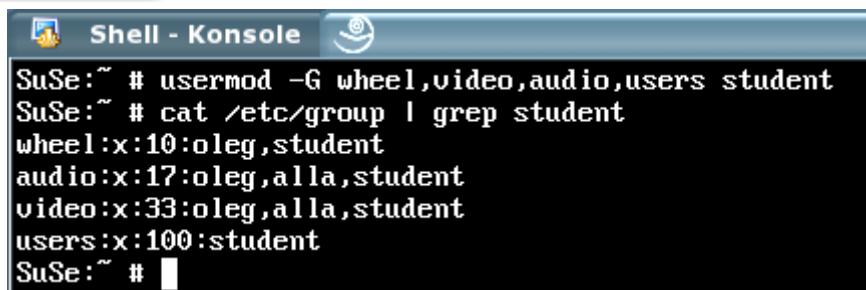


```
Shell No. 3 - Konsole
Session Edit View Bookmarks Settings Help

oleg@SuSe:~> ls -la /etc/skel/
total 80
drwxr-xr-x  7 root  root      568 2004-03-20 12:29 .
drwxr-xr-x  77 root  root    7664 2004-06-03 00:02 ..
-rw-----  1 root  root       0 1996-05-18 18:20 .bash_history
-rw-r--r--  1 root  root   1286 2002-07-30 13:33 .bashrc
drwxr-xr-x  2 root  root      48 2003-09-02 13:00 bin
drwxr-xr-x  2 root  root     80 2004-03-20 12:12 Documents
-rw-r--r--  1 root  root   208 1995-11-17 20:22 .dvipsrc
-rw-r--r--  1 root  root  1637 2002-01-25 15:37 .emacs
-rw-r--r--  1 root  root  1124 2000-02-28 23:05 .exrc
drwxr-xr-x  2 root  root      48 2003-03-15 20:40 .fonts
-r--r--r--  1 root  root  16035 2003-09-24 00:01 .gnu-emacs
-rw-r--r--  1 root  root    164 1995-11-17 20:22 .kermrc
-rw-r--r--  1 root  root   6148 2003-09-17 23:15 .muttrc
-rw-r--r--  1 root  root    934 2002-07-17 14:42 .profile
drwxr-xr-x  2 root  root     80 2004-03-20 12:12 public_html
-rw-r--r--  1 root  root    311 2000-07-07 15:55 .urlview
-rw-r--r--  1 root  root   7913 1995-11-30 18:48 .xcoralrc
drwxr-xr-x  2 root  root     72 2004-03-20 12:12 .xemacs
-rw-r--r--  1 root  root   4027 2003-08-06 17:03 .xim.template
-rwrxr-xr-x  1 root  root   2980 2003-04-13 00:25 .xinitrc.template
-rw-r--r--  1 root  root    119 1997-10-28 14:39 .xtalkrc
oleg@SuSe:~>
```

Для того, чтобы изменить какие-либо параметры, у уже существующего пользователя, используется команда `usermod`. В частности, попробуем изменить группы, в которые входит пользователь:





```
SuSe:~ # usermod -G wheel,video,audio,users student
SuSe:~ # cat /etc/group | grep student
wheel:x:10:oleg,student
audio:x:17:oleg,alla,student
video:x:33:oleg,alla,student
users:x:100:student
SuSe:~ #
```

Вы в состоянии самостоятельно определить, какие именно каталоги и файлы необходимо размещать сразу после добавления пользователя в его домашний каталог. При этом вы должны продумать содержимое каждого файла, так как зачастую это позволяет нам сэкономить очень много времени для дополнительной настройки системы для данного пользователя.

В частности, в файле *.bash\_history* хранится история команд, вводимых в консоли тем пользователем, в чьем домашнем каталоге мы просматриваем данный файл. История сохраняется не только за текущий сеанс, но и за многие предыдущие. Это облегчает пользователю вводить команды, синтаксис которых очень непрост, но однажды данному пользователю уже удалось разобраться в тонкостях применения данной команды.

В файле *.bashrc* можно разместить какие-либо команды, которые будут выполняться автоматически при входе пользователя в консоль. Некий аналог пользовательского сценария Windows.

Обратите внимание, что для удобства пользователя могут быть сразу созданы каталоги, где он будет сохранять свои документы: *Documents*, а также где он будет размещать свои разработанные интернет-странички: *public\_html*. Здесь же уже могут храниться файлы с готовыми настройками различных приложений. Все подобные файлы считаются "скрытыми", их название начинается со знака точки.

Для создания группы необходимо воспользоваться командой *groupadd имя\_группы*

Для удаления пользователя используется команда *userdel*, для удаления группы *groupdel*.

- Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава
- Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков



## 4.5 Модификация временных параметров пользователя

Для модификации временных параметров учетной записи, так же как и для модификации остальных характеристик, можно, например, воспользоваться простым редактированием файла паролей. Однако ввиду сложностей с "летоисчислением от рождества UNIX", использование утилиты chage (change age) является более предпочтительным средством. Перечислим возможные ключи данной утилиты:

*chage -m (minimum)* - определяет значение поля №4 файла /etc/shadow

*chage -M (Maximum)* - определяет значение поля №5 файла /etc/shadow

*chage -W (Warning)* - определяет значение поля №6 файла /etc/shadow

*chage -I (Inactive)* - определяет значение поля №7 файла /etc/shadow

*chage -E (Expired)* - определяет значение поля №8 файла /etc/shadow

## 4.6 Задания для самопроверки

1. Создайте пользователя *student* и задайте ему пароль. Потребуйте от пользователя обязательную смену пароля при первом входе в систему.
2. Настройте систему таким образом, чтобы у всех новых пользователей, чьи учетные записи будут создаваться в будущем, на рабочем столе находился документ с правилами безопасности вашей организации.
3. Создайте группу *interns*. Добавьте туда пользователя *student*. Настройте систему так, чтобы данный пользователь не смог пользоваться системой после окончания своей практики на вашем предприятии. Срок окончания – 1 января, 2013 года.
4. Создайте учетную запись пользователя *poweroff* и настройте ее таким образом, чтобы при входе данного пользователя в систему, производилось автоматическое выключение компьютера.
5. Настройте систему таким образом, чтобы пользователь *student* не мог заходить в систему, если пользователь *root* входил в систему и не выходил из нее.



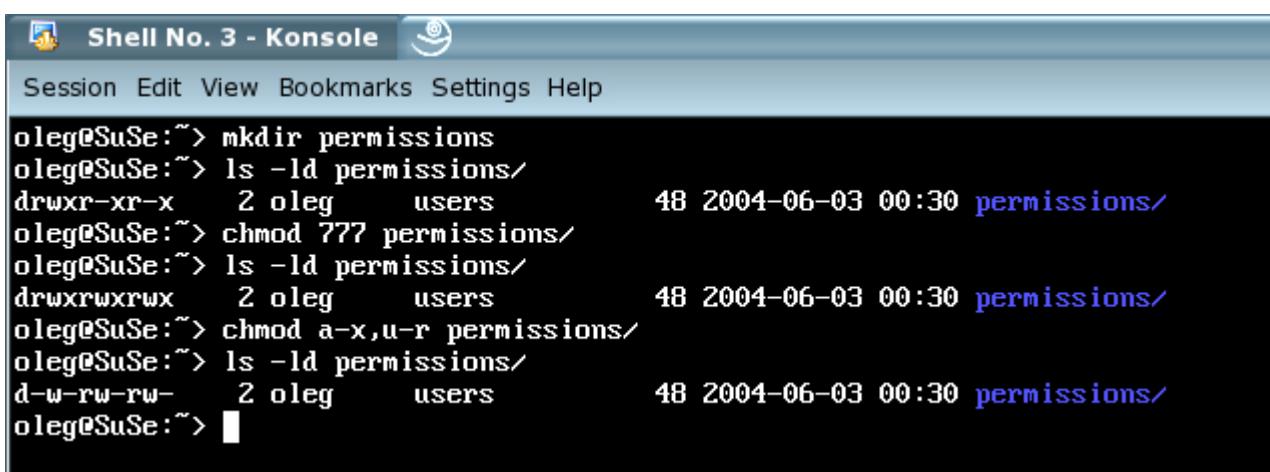


## 5. Права доступа к файлам и каталогам

Разработка единых правил доступа к файлам и каталогам - чрезвычайно важный шаг в защите вычислительной среды. Многие бреши возникают лишь из-за наличия небезопасных прав доступа к тому или иному файлу или каталогу. В этом разделе будет рассмотрено, какие существуют права доступа к файлам и каталогам и что они означают.

### 5.1 Просмотр прав доступа

Команда *ls -l*, помимо всего прочего, отображает информацию о типе файла и правах доступа к нему. Создадим папку, определим установленные на нее права, попробуем их изменить:



```
Shell No. 3 - Konsole
Session Edit View Bookmarks Settings Help
oleg@SuSe:~> mkdir permissions
oleg@SuSe:~> ls -ld permissions/
drwxr-xr-x  2 oleg    users            48 2004-06-03 00:30 permissions/
oleg@SuSe:~> chmod 777 permissions/
oleg@SuSe:~> ls -ld permissions/
drwxrwxrwx  2 oleg    users            48 2004-06-03 00:30 permissions/
oleg@SuSe:~> chmod a-x,u-r permissions/
oleg@SuSe:~> ls -ld permissions/
d-w-rw-rw-  2 oleg    users            48 2004-06-03 00:30 permissions/
oleg@SuSe:~> █
```

После выполнения команды *ls -ld* нам была продемонстрирована следующая информация:

*drwxr-xr-x*

Первая буква “*d*” говорит о том, что данный объект - каталог. Информация о правах доступа отображается в виде триплетов. Первый триплет относится к владельцу файла, второй к группе, которой принадлежит файл, третий к прочим пользователям. Каждый триплет состоит из 3 полей (3 букв, символизирующих определенное право доступа к данному файлу). Расшифровка букв такова:

- Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава
- Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков





#### **r - чтение.**

Для файла это означает доступность для чтения, для каталога - возможность просмотра содержимого. Этот символ может стоять в левом поле триплета

#### **w - запись.**

Для файла это означает возможность записи, для каталога - возможность удаления и добавления файлов. Этот символ может стоять в центральном поле триплета.

#### **x - выполнение.**

Для файла это означает возможность выполнения, для каталога - возможность поиска информации в нем. Этот символ может стоять в правом поле триплета.

#### **s - бит SUID или SGID.**

Этот символ может стоять в правом поле первого (владелец) или второго (группа) триплета. В первом случае это означает, что при выполнении файл примет идентификатор владельца, а не пользователя, запустившего программу. Во втором случае это означает, что при выполнении файл примет идентификатор группы, которой он принадлежит, а не первичный идентификатор группы того пользователя, который запустил программу. Если бит SGID задан для каталога, то все новые файлы и подкаталоги унаследуют его идентификатор группы, а не значение GID процесса, создающего файл или подкаталог.

#### **t - sticky-бит.**

Этот символ может стоять в правом поле третьего (остальные пользователи) триплета некоторых каталогов, например /var/tmp. Обычно, если в третьем триплете для каталога задано право записи и выполнения, то любой пользователь сможет удалить любой файл из этого каталога независимо от того, кому принадлежит этот файл и какие у него права доступа. Специальный sticky-бит разрешает удаление и модификацию файлов только владельцам и суперпользователю.

#### **-- нет доступа.**

Этот символ может стоять в любом поле любого триплета и означает запрет соответствующего права доступа.





Права доступа также удобно представлять в виде восьмеричных значений. Правому полю каждого триплета соответствует значение 1, центральному 2, левому 4. При указании таким образом прав, необходимо помнить о "специальных" правах доступа (*SUID SGID sticky-bit*). Этот триплет находится первым слева, 4 - соответствует биту *SUID*, 2 - *SGID*, 1 - *sticky-bit*.

## 5.2 Изменение прав доступа

На предыдущей иллюстрации, для изменения прав доступа была применена команда *chmod*. Командой *chmod 777 имя\_файла* мы включили все биты доступа, за исключением специальных, что отражается последующей командой просмотра. Помимо восьмеричного представления прав доступа, есть возможность использования буквенных обозначений. Для этого используются буквенные сокращения различных триплетов:

- a* - all. Все пользователи
- u* - user. Пользователь - владелец.
- g* - group. Группа владельцев.
- o* - other. Все остальные.

Итак, команда *chmod a-x, u+r имя\_файла* - позволяет снять бит выполнения для всех пользователей и чтения для владельца. Если использовать знак + вместо -, то это будет означать установку указанного бита. Также можно использовать и знак «=», точно указывая, какие права доступа необходимо установить.

Каталог с такими правами будет недоступен для входа в него всеми пользователями, а также владельцу каталога не будет доступен просмотр его содержимого. Следует также помнить, что пользователь *root* всегда имеет доступ к файлам или каталогам, независимо от того, какие права доступа для данных объектов установлены.

Однако, будучи владельцем каталога, мы всегда можем изменить права доступа к нему, что позволено только владельцу и пользователю *root*. Поэтому, вернув себе право на выполнение - мы обеспечим возможность входа в каталог, а вернув право на чтение - возможность просмотра. Изменим права доступа выше созданного каталога таким образом, чтобы пользователи имели возможность просмотра его содержимого:





Shell No. 3 - Konsole

Session Edit View Bookmarks Settings Help

```
oleg@SuSe:~> cd permissions/
bash: cd: permissions/: Permission denied
oleg@SuSe:~> chmod u+x permissions/
oleg@SuSe:~> cd permissions/
oleg@SuSe:~/permissions> touch 1.txt
oleg@SuSe:~/permissions> ls -la
/bin/ls: ..: Permission denied
oleg@SuSe:~/permissions> chmod u+r ..
oleg@SuSe:~/permissions> ls -la
/bin/ls: ..: Permission denied
oleg@SuSe:~/permissions> chmod u+r .
oleg@SuSe:~/permissions> ls -la
total 6
drwxrwx-rw- 2 oleg      users          72 2004-06-03 00:33 .
drwxr-xr-x 100 oleg      wheel        6592 2004-06-03 00:31 ..
-rw-r--r--  1 oleg      users          0 2004-06-03 00:33 1.txt
oleg@SuSe:~/permissions>
```

Обратите внимание на то, что, имея только право на запись, и не имея прав на вход и просмотр, нам, тем не менее, удалось создать файл в целевом каталоге.

### 5.3 Владение файлом или каталогом

В третьем и четвертом полях, выводимых командой *ls*, указывается имя пользователя - владельца данного объекта, и название группы владельцев данного объекта. При создании файла или каталога, владельцем данного объекта становится тот пользователь, который создавал объект, а группой владельцев - основная группа данного пользователя.

Команда *chown* позволяет изменить владельца файла или каталога. Проводить подобную операцию разрешается суперпользователю и текущему владельцу объекта. Командой *chgrp* мы имеем возможность сменить группу владельцев указанного объекта.

Рассмотрим пример, который иллюстрирует изменение владельца каталога */home/oLeg/permissions* с пользователя *oLeg* на пользователя *student*, а затем группу владельцев *users* на группу *wheel*. При этом сам пользователь *student* может не являться членом группы *wheel*:



Shell No. 3 - Konsole

Session Edit View Bookmarks Settings Help

```
SuSe:~ # ls -la /home/oleg/permissions/
total 6
drwxrwx-rw- 3 oleg users 96 Jun 3 00:36 .
drwxr-xr-x 100 oleg wheel 6592 Jun 3 00:38 ..
-rw-r--r-- 1 oleg users 0 Jun 3 00:33 1.txt
drwxr-xr-x 2 oleg users 48 Jun 3 00:36 new
SuSe:~ # chown student /home/oleg/permissions/
SuSe:~ # ls -la /home/oleg/permissions/
total 6
drwxrwx-rw- 3 student users 96 Jun 3 00:36 .
drwxr-xr-x 100 oleg wheel 6592 Jun 3 00:38 ..
-rw-r--r-- 1 oleg users 0 Jun 3 00:33 1.txt
drwxr-xr-x 2 oleg users 48 Jun 3 00:36 new
SuSe:~ # chgrp wheel /home/oleg/permissions/
SuSe:~ # ls -la /home/oleg/permissions/
total 6
drwxrwx-rw- 3 student wheel 96 Jun 3 00:36 .
drwxr-xr-x 100 oleg wheel 6592 Jun 3 00:38 ..
-rw-r--r-- 1 oleg users 0 Jun 3 00:33 1.txt
drwxr-xr-x 2 oleg users 48 Jun 3 00:36 new
SuSe:~ # █
```

## 5.4 SUID и SGID

Как уже было отмечено, зачастую необходимо выполнять то или иное приложение от имени его владельца. Напомню, что запустить можно лишь то приложение, для которого у нас есть разрешение на его запуск, то есть бит x в необходимом триплете.

Однако не всегда запуск приложения проходит успешно, даже если у нас есть право на его запуск. Это происходит потому, что запускается приложение от имени определенного пользователя, точнее от его номера-идентификатора.

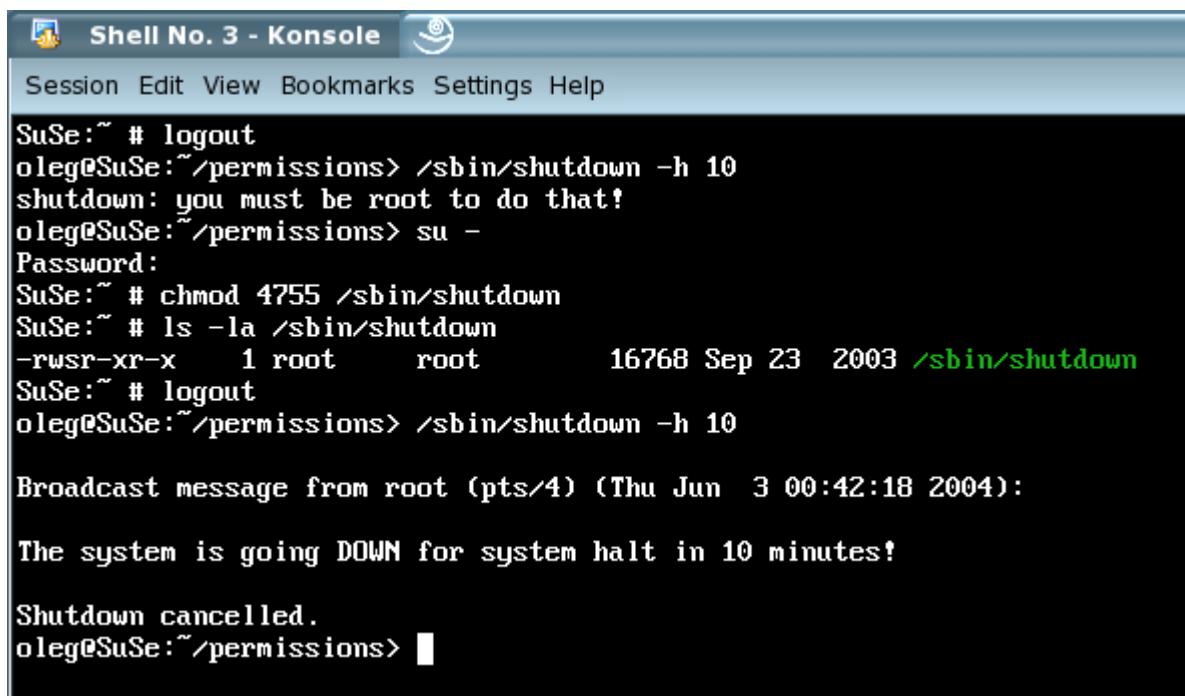
Не все приложения могут выполнить определенные действия от имени любого пользователя. Например, при выполнении команды *shutdown* от имени непrivилегированного пользователя, система отказывается выполнять выключение, мотивируя тем, что мы должны обладать правами суперпользователя для выполнения столь важной операции.

Установив бит SUID, мы даем инструкцию на то, что при запуске *shutdown*, система должна «думать», что на самом деле эту утилиту запустил ее владелец - суперпользователь. Таким нехитрым способом можно разрешить пользователям

- |          |                  |            |             |             |           |
|----------|------------------|------------|-------------|-------------|-----------|
| ■ Киев   | ■ Днепропетровск | ■ Львов    | ■ Ровно     | ■ Мариуполь | ■ Полтава |
| ■ Одесса | ■ Донецк         | ■ Николаев | ■ Запорожье | ■ Луганск   | ■ Харьков |



монтировать каталоги, дозваниваться к провайдеру и т.д. Однако, необходимо помнить о том, что всякая программа с битами *SUID* и *SGID* - потенциальная брешь в системе безопасности. По возможности, необходимо избегать использования данных бит доступа.



```
Shell No. 3 - Konsole
Session Edit View Bookmarks Settings Help

SuSe:~ # logout
oleg@SuSe:~/permissions> /sbin/shutdown -h 10
shutdown: you must be root to do that!
oleg@SuSe:~/permissions> su -
Password:
SuSe:~ # chmod 4755 /sbin/shutdown
SuSe:~ # ls -la /sbin/shutdown
-rwsr-xr-x  1 root      root      16768 Sep 23  2003 /sbin/shutdown
SuSe:~ # logout
oleg@SuSe:~/permissions> /sbin/shutdown -h 10

Broadcast message from root (pts/4) (Thu Jun  3 00:42:18 2004):
The system is going DOWN for system halt in 10 minutes!

Shutdown cancelled.
oleg@SuSe:~/permissions>
```

## 5.5 Sticky – бит

Создадим каталог, под названием *sticky*, в котором будут иметь возможность различные пользователи создавать свои каталоги, а также заходить в данный каталог. Для этого необходимо установить права *w* и *x* для всех пользователей. Далее, создадим в нем два каталога, от имени разных пользователей. Установим права на эти созданные каталоги таким образом, чтобы прав на запись, а значит и удаление данных каталогов, у "не владельцев" не было.

В таком случае, руководствуясь логикой, можно предположить, что пользователь, не являющийся владельцем каталога, не сможет удалить «чужой» каталог. Проверим это на практике:





Shell No. 2 - Konsole

Session Edit View Bookmarks Settings Help

```

oleg@SuSe:~> mkdir sticky
oleg@SuSe:~> ls -ld sticky/
drwxr-xr-x    2 oleg      users            48 2004-06-04 03:09 sticky/
oleg@SuSe:~> chmod 777 sticky/
oleg@SuSe:~> mkdir sticky/oleg
oleg@SuSe:~> su - student
Password:
su: warning: cannot change directory to /home/student: No such file or directory
student@SuSe:/home/oleg> mkdir sticky/student
student@SuSe:/home/oleg> ls sticky/
oleg  student
student@SuSe:/home/oleg> ls -la sticky/
total 7
drwxrwxrwx    4 oleg      users            96 2004-06-04 03:10 .
drwxr-xr-x   103 oleg     wheel           6696 2004-06-04 03:09 ..
drwxr-xr-x    2 oleg      users            48 2004-06-04 03:09 oleg
drwxr-xr-x    2 student   audio           48 2004-06-04 03:10 student
student@SuSe:/home/oleg> rm -r sticky/oleg
rm: remove write-protected directory `sticky/oleg'? y
student@SuSe:/home/oleg> ls -la sticky/
total 7
drwxrwxrwx    3 oleg      users            72 2004-06-04 03:11 .
drwxr-xr-x   103 oleg     wheel           6696 2004-06-04 03:09 ..
drwxr-xr-x    2 student   audio           48 2004-06-04 03:10 student
student@SuSe:/home/oleg>

```

Однако ввиду того, что на родительский каталог установлены полные права доступа, пользователь, не являющийся владельцем каталога и не имеющего полные права на него, все-таки удаляет каталог другого пользователя. При этом система просит уточнить, действительно ли нужно удалить каталог, который доступен пользователю только для чтения.

Попробуем исправить данную ситуацию (если она нас действительно не устраивает). Для этого необходимо установить на родительский каталог *sticky-bit*.

*Sticky-bit* выступает в роли "миротворца", дабы не разжигать войн между пользователями, удалившими не свой каталог или его содержимое. Таким образом, если в какой-то каталог у вас установлен полный доступ, но вы хотите защитить файлы и каталоги, находящиеся внутри данного каталога от посягательств "не владельцев" - установите на родительский для них каталог *sticky-bit*!





```

Shell No. 2 - Konsole
Session Edit View Bookmarks Settings Help

oleg@SuSe:~> ls -la sticky/
total 7
drwxrwxrwx 3 oleg users 72 2004-06-04 03:11 .
drwxr-xr-x 103 oleg wheel 6696 2004-06-04 03:12 ..
drwxr-xr-x 2 student audio 48 2004-06-04 03:10 student
oleg@SuSe:~> chmod 1777 sticky/
oleg@SuSe:~> ls -la sticky/
total 7
drwxrwxrwt 3 oleg users 72 2004-06-04 03:11 .
drwxr-xr-x 103 oleg wheel 6696 2004-06-04 03:12 ..
drwxr-xr-x 2 student audio 48 2004-06-04 03:10 student
oleg@SuSe:~> mkdir sticky/oleg
oleg@SuSe:~> su - student
Password:
su: warning: cannot change directory to /home/student: No such file or directory
student@SuSe:/home/oleg> ls -la sticky/
total 7
drwxrwxrwt 4 oleg users 96 2004-06-04 03:18 .
drwxr-xr-x 103 oleg wheel 6696 2004-06-04 03:12 ..
drwxr-xr-x 2 oleg users 48 2004-06-04 03:18 oleg
drwxr-xr-x 2 student audio 48 2004-06-04 03:10 student
student@SuSe:/home/oleg> rm -r sticky/oleg/
rm: remove write-protected directory `sticky/oleg/'? y
rm: remove write-protected directory `sticky/oleg/'? y
rm: cannot remove directory `sticky/oleg/': Operation not permitted
student@SuSe:/home/oleg>

```

## 5.6 Задания для самопроверки

- С помощью утилиты *find* составьте список всех файлов с установленными битами *SUID* и *SGID*. Проанализируйте полученный список.
- Настройте систему таким образом, чтобы игру *frozen-bubble* мог запускать только пользователь *gamer*.
- Настройте систему таким образом, чтобы из непrivилегированных пользователей только пользователь *poweroff* мог выключать компьютер командой *shutdown*. Выполните это задание без изменения *UID* данного пользователя в файле */etc/passwd*.
- Разрешите пользователю *admin* создавать учетные записи пользователей.



## 6. Делегирование прав суперпользователя

В большой организации, в которой работает целый штат администраторов, появляется необходимость в распределении обязанностей между ними. Пароль от учетной записи *root* должен быть известен только одному администратору, который с помощью утилиты *sudo* может делегировать свои полномочия конкретным пользователям или группам.

### 6.1 Sudo

Утилита *sudo* позволяет предоставлять рядовым пользователям отдельные права суперпользователя и регистрировать с помощью службы аудита событий факт выполнения каждой привилегированной команды. Это может происходить на уровне отдельных систем или всей сети в целом. Данным инструментом можно настраивать "группы пользователей по интересам". Например, группу "Операторы Архива" или группу "Операторы настройки сети".

Списки привилегированных пользователей и разрешенных команд приведены в конфигурационном файле */etc/sudoers*. Когда пользователь вызывает утилиту *sudo*, отображается запрос на ввод пароля. Если пользователь успешно прошел аутентификацию, утилита выполняет требуемую команду и завершается. В течение последующих пяти минут пользователь может запросить дополнительные команды и даже запустить интерпретатор команд в режиме суперпользователя, не вводя пароль (5 минут - это установка по умолчанию - задается на этапе компиляции программы).

При каждом вызове утилиты *sudo* служба аудита фиксирует выполняемую команду вместе с флагами и аргументами командной строки. Запись в журнальном файле содержит дату и метку времени, имя пользователя, название терминала, имя текущего каталога, а также сообщение об ошибке, если таковое имеется.

В современных системах по соображениям безопасности не рекомендуется работать под учетной записью суперпользователя. В некоторых дистрибутивах Linux под данной учетной записью невозможно выполнить вход в систему и все действия, для которых требуются повышенные привилегии, необходимо выполнять с помощью *sudo*.

По умолчанию, как правило, *sudo* сконфигурирована таким образом, что любой пользователь может выполнять любую команду, если он знает пароль «целевого»

- █ Киев    █ Днепропетровск    █ Львов    █ Ровно    █ Мариуполь    █ Полтава
- █ Одесса    █ Донецк    █ Николаев    █ Запорожье    █ Луганск    █ Харьков





пользователя, то есть пароль того пользователя, от имени которого он желает запустить команду. Как правило, целевым пользователем является пользователь *root*. За такой режим работы отвечают две следующие опции файла */etc/sudoers*:

```
Defaults targetpw
ALL  ALL=(ALL)  ALL
```

Первая строка определяет такой режим, при котором у пользователей, которые пользуются *sudo*, запрашивается не их личный пароль, а пароль целевого пользователя. Вторая строка соответствует следующему синтаксису:

```
пользователь имя_хоста_пользователя=(целевой_пользователь) команда
```

Таким образом, *ALL ALL=(ALL) ALL*, означает, что любой пользователь, зашедший в систему с локального или любого другого удаленного компьютера, может выполнять от имени любого целевого пользователя, любые команды. Безусловно, эта опция должна применяться только в паре с опцией *Defaults targetpw*.

Для того, чтобы использовать *sudo* в режиме, при котором пользователи пользуются командами, указывая собственный пароль, необходимо удалить или закомментировать две вышеуказанные нами строки файла */etc/sudoers*. Рассмотрим пример конфигурации:

```
# In the default (unconfigured) configuration, sudo asks for the root password.
# This allows use of an ordinary user account for administration of a freshly
# installed system. When configuring sudo, delete the two
# following lines:
#Defaults targetpw  # ask for the password of the target user i.e. root
#ALL  ALL=(ALL) ALL  # WARNING! Only use this together with 'Defaults targetpw'!

# Runas alias specification

# User privilege specification
root    ALL=(ALL) ALL

# Uncomment to allow people in group wheel to run all commands
# %wheel      ALL=(ALL) ALL
oleg    ALL=(ALL) /usr/sbin/useradd usr_*
# Same thing without a password
# %wheel      ALL=(ALL) NOPASSWD: ALL
```



В рассматриваемом файле */etc/sudoers* пользователю *oleg* разрешено создавать пользователей, имя которых начинается с префикса *usr\_*. Пользователю *root* разрешено выполнять любые команды от имени любых пользователей. Воспользуемся предоставленными привилегиями:

```
oleg@mail:~> /usr/sbin/useradd tux
Cannot lock password file: already locked.
oleg@mail:~> sudo /usr/sbin/useradd tux
Sorry, user oleg is not allowed to execute '/usr/sbin/useradd tux' as root on mail.
oleg@mail:~> sudo /usr/sbin/useradd usr_tux
oleg@mail:~> _
```

Первой командой пользователь *oleg* безуспешно попытался создать пользователя по имени *tux*. Используя утилиту *sudo* данную операцию также не удалось выполнить по той причине, что *oleg* имеет право создавать только таких пользователей, у которых в имени присутствует префикс *usr\_*. Третья команда не вызвала у системы каких-либо нареканий, и она была беспрекословно выполнена.

Конфигурационный файл */etc/sudoers* поддерживает специальные псевдонимы, которые позволяют более рационально организовать конфигурацию данного файла, если в нем требуется достаточно большое количество записей. Рассмотрим фрагменты файла */etc/sudoers*:

```
# User alias specification
User_Alias ADMINS = oleg, test

# Cmnd alias specification

Cmnd_Alias USERCMD = /usr/sbin/useradd usr_*, /usr/bin/passwd usr_*

#oleg  ALL=(ALL) /usr/sbin/useradd usr_*
ADMINS  ALL=(ALL)          USERCMD
```

В первой незакомментированной строке создается псевдоним *ADMINS*, который хранит в себе список пользователей (*oleg, test*). Вторая строка создает псевдоним *USERCMD*, в котором перечисляются команды создания учетных записей и установки им пароля. В третьей строке указано, что пользователи из псевдонима *ADMINS* могут выполнять команды, указанные в псевдониме *USERCMD*. Такая организация файла */etc/sudoers* позволит администратору быстрее и проще назначать необходимые привилегии определенным пользователям.



В качестве псевдонимов в файле */etc/sudoers* могут выступать:

Псевдоним	Описание
<i>Host_Alias</i>	Список разделенных запятыми имен и/или IP-адресов хостов
<i>Runas_Alias</i>	Список разделенных запятыми пользователей, которых можно указывать командой <i>sudo -u</i> (целевые пользователи)
<i>Cmnd_Alias</i>	Список разделенных запятыми команд. Если указывается каталог, то разрешены все команды из данного каталога.
<i>User_Alias</i>	Список разделенных запятыми пользователей и групп

В файле */etc/sudoers* допускается использовать различные метасимволы:

Псевдоним	Описание
*	Произвольное число символов
?	Один произвольный символ
[диапазон, диапазон,...]	Любой символ диапазона
[!диапазон, !диапазон,...]	Любой символ, не входящий в диапазон
\	Отмена специальной интерпретации следующего символа
#	Признак комментария. Если это UID пользователя, то берется в кавычки “#”
%	После этого знака следует указывать имя группы пользователей

## 6.2 Задание для самопроверки

1. Настройте систему таким образом, чтобы пользователь *poweroff* мог выключать ее, но с обязательным условием: пользователи, работающие в системе, должны иметь 5 минут на то, чтобы закончить свою работу перед выключением.
2. Создайте группу *admins* и разрешите пользователям этой группы выполнение всех административных программ.



## 7. Подключаемые модули аутентификации

Подключаемые модули аутентификации (Pluggable Authentication Module — PAM) не защищают систему после того, как она была взломана, но они могут помочь предотвратить взлом. Повышение безопасности достигается за счет чрезвычайно гибкой схемы идентификации. В частности, в традиционной схеме пользователи UNIX аутентифицируются, предоставляя пароль в командной строке после ввода регистрационного имени в приглашении *Login*. Во многих случаях, например при локальном доступе к рабочим ресурсам, этого оказывается вполне достаточно. Но иногда требуется дополнительная информация.

Когда пользователь регистрируется в системе через внешний канал, такой как Internet, может понадобиться дополнить или поменять традиционный механизм аутентификации. Все это реализуется с помощью модулей PAM. Самое главное, они позволяют конфигурировать вычислительную среду в соответствии с требуемым уровнем безопасности. Поддержка модулей PAM есть во всех новых дистрибутивах Linux. Если же по какой-то причине она отсутствует, обратитесь по следующему адресу: <http://www.kernel.org/pub/Linux/Libs/pam>. Там можно найти исходные коды и документацию.

### 7.1 Алгоритм работы модулей

PAM - это центральный механизм аутентификации всех служб, в том числе команды *Login*, утилит дистанционной регистрации (*telnet*, *rlogin*, *rsh*, *ftp*) протокола PPP и команды *su*. Модули PAM позволяют ограничить доступ к приложениям и время доступа к системе, реализовать альтернативные схемы аутентификации, осуществить расширенную регистрацию событий и многое другое. Они могут использоваться с любыми приложениями Linux.

В общем случае их работу можно описать следующим алгоритмом:

1. Приложение, например *Login*, делает первичное обращение к Linux-PAM.
2. Ядро Linux-PAM находит соответствующий конфигурационный файл в каталоге */etc/pam.d* (или обращается к файлу */etc/pam.conf*) и загружает из него список модулей, необходимых для обслуживания запроса.



3. После этого ядро Linux-PAM загружает модули в том порядке, в котором они перечислены в конфигурационном файле. Не все модули нужно загружать - это зависит от параметров конфигурации.
4. Некоторые модули организуют диалог с пользователем через вызывающее приложение. В ходе диалога обычно выдается приглашение на ввод различных параметров, например пароля. Если предоставленные пользователем данные подходят, то управление возвращается ядру Linux-PAM, которое вызывает следующий модуль.

В конечном итоге процедура аутентификации заканчивается успехом или неудачей. В случае неудачи выдается обобщенное сообщение об ошибке, которое, как правило, не раскрывает причину такого результата. Это своего рода защитная мера, злоумышленники, пытающиеся взломать систему, не смогли узнать ничего лишнего. В то же время, все модули оснащены средствами журнальной регистрации, что позволяет системным администраторам выявлять источники проблем и попытки нарушения прав доступа.

## 7.2 Конфигурирование

Существует два конфигурационных параметра компиляции модулей PAM. Первый из них заставляет собой модуль использовать либо файл `/etc/pam.conf`, либо группу файлов в каталоге `/etc/pam.d`. Второй подключает оба конфигурационных механизма, причем записи, найденные в каталоге `/etc/pam.d`, имеют приоритет над записями файла `/etc/pam.conf`. Рекомендуется придерживаться первого варианта, именно он реализован по умолчанию во многих дистрибутивах.

Формат файла `/etc/pam.conf` и файлов каталога `/etc/pam.d` неодинаков. В первом случае имеется начальное поле, определяющее тип службы, т.е. приложение, к которому относится данная запись. Во втором случае в каталоге находятся файлы, имя каждого из которых соответствует названию приложения. Соответственно, поле типа службы в этих файлах отсутствует. Мы будем рассматривать формат конфигурационных файлов, предполагая, что они находятся в каталоге `/etc/pam.d`.

Начнем со знакомства с содержимым каталога `/etc/pam.d`. В данном каталоге перечислены все PAM - приложения входящие в используемый Вами дистрибутив.





## Терминал

Файл Правка Вид Терминал Вкладки Справка

mail:~/Desktop # ls /etc/pam.d/

atd	common-session.pam-config-backup	openwsman	squid
chage	common-session.pc	other	sshd
chfn	crond	passwd	su
chsh	cups	polkit	sudo
<b>common-account</b>	gdm	polkit-1	su-l
common-account.pam-config-backup	gdm-autologin	pop3	useradd
common-account.pc	gnome-passwd	ppp	vlock
<b>common-auth</b>	gnome-screensaver	pure-ftpd	xdm
common-auth.pam-config-backup	gnome-screensaver-smartcard	quagga	xdm-np
common-auth.pc	gnomesu-pam	rpasswd	xen-api
<b>common-password</b>	imap	samba	xlock
common-password.pam-config-backup	kcheckpass	sfcf	xscreensaver
common-password.pc	login	shadow	
<b>common-session</b>	login.old	smtp	

Формат содержимого файлов таков:

*Тип\_модуля      управляющий\_флаг      путь\_к\_модулю      аргументы*

Типы модулей:

Тип	Описание
<i>Auth</i>	Модуль данного типа заставляет приложение запрашивать у пользователя данные для аутентификации, задает права доступа и предоставляет привилегии.
<i>account</i>	Модуль данного типа проверяет различные параметры учетной записи пользователя. Например, устаревание пароля, доступ к системе в определенное время работы и т.д.
<i>session</i>	Модуль данного типа используется для выполнения определенных действий до и после организации сеанса работы.
<i>password</i>	Модуль данного типа отвечает за обновление пользовательского маркера аутентификации (как правило, это пароль пользователя)

Управляющие флаги:

Тип	Описание
<i>required</i>	Работа модуля должна выполниться успешно, чтобы пользователь получил доступ к службе. Если модуль включен в стек, остальные модули также выполняются.

- Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава
- Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков





Приложению не сообщается о том, какой из модулей не был пройден пользователем.

*requisite*

Аналогично предыдущему, но в случае неудачи, оставшиеся модули в стеке не выполняются. Приложению немедленно сообщается об ошибке.

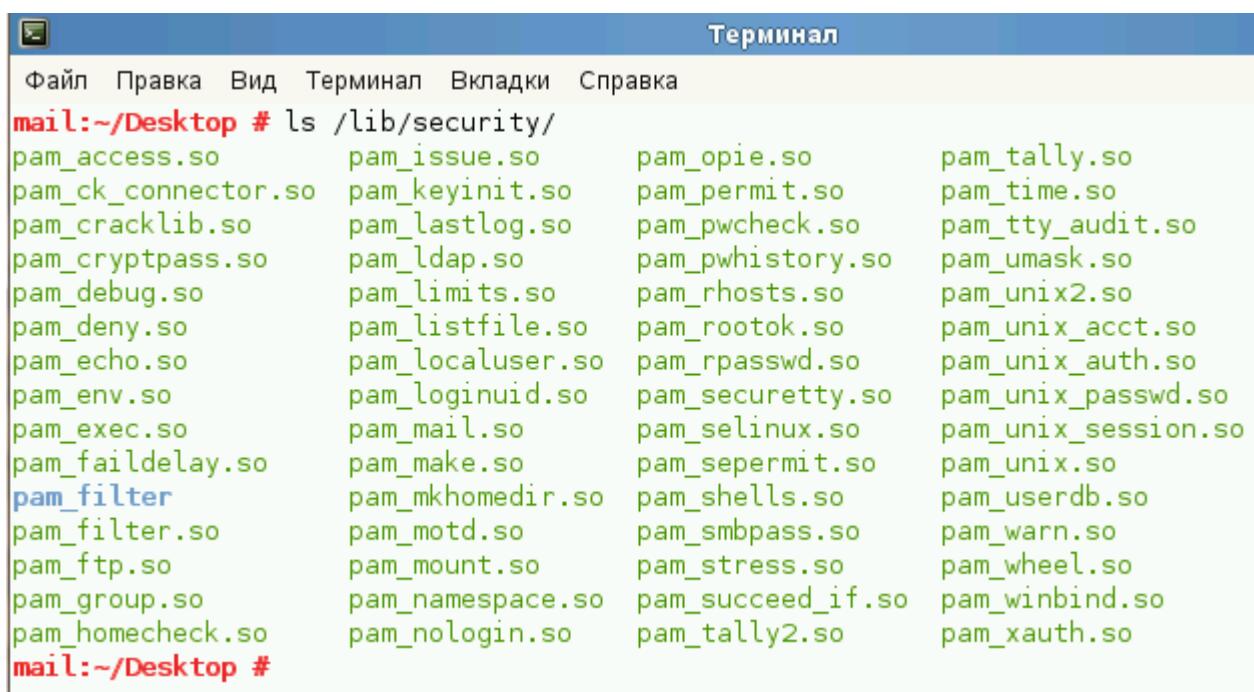
*sufficient*

Если работа модуля успешно выполняется, то все остальные модули в стеке игнорируются. Приложению возвращается код успешной аутентификации. Если модуль в стеке не единственный, то неуспешное его выполнение не сказывается отрицательно на прохождение стека

*optional*

Необязательный модуль, но если он единственный в стеке, в случае ошибки возвращается код ошибки

Все модули, как правило, хранятся в каталоге */lib/security*



```
Файл Правка Вид Терминал Вкладки Справка
mail:~/Desktop # ls /lib/security/
pam_access.so      pam_issue.so      pam_opie.so      pam_tally.so
pam_ck_connector.so pam_keyinit.so    pam_permit.so   pam_time.so
pam_cracklib.so   pam_lastlog.so   pam_pwcheck.so pam_tty_audit.so
pam_cryptpass.so   pam_ldap.so     pam_pwhistory.so pam_umask.so
pam_debug.so       pam_limits.so    pam_rhosts.so   pam_unix2.so
pam_deny.so        pam_listfile.so  pam_rootok.so   pam_unix_acct.so
pam_echo.so        pam_localuser.so pam_rpasswd.so pam_unix_auth.so
pam_env.so         pam_loginuid.so  pam_securetty.so pam_unix_passwd.so
pam_exec.so        pam_mail.so     pam_selinux.so  pam_unix_session.so
pam_failedelay.so  pam_make.so     pam_sepermit.so pam_unix.so
pam_filter         pam_mkhomedir.so pam_shells.so   pam_userdb.so
pam_filter.so      pam_motd.so    pam_smbpass.so  pam_warn.so
pam_ftp.so         pam_mount.so   pam_stress.so   pam_wheel.so
pam_group.so       pam_namespace.so pam_succeed_if.so pam_winbind.so
pam_homecheck.so   pam_nologin.so  pam_tally2.so   pam_xauth.so
mail:~/Desktop #
```

В конфигурационных файлах, находящихся в каталоге */etc/pam.d* можно указывать как полный путь к файлу модуля, так и относительный путь (относительно каталога */lib/security*).

Итак, каждый файл в каталоге */etc/pam.d* связан с определенным приложением, по имени которого он назван. Этот файл содержит список записей, в каждой из которых задается тип модуля, управляющий флаг, путь к модулю и дополнительные аргументы. Модули одного типа считаются объединенными в стек и выполняются в указанном

- Киев
- Днепропетровск
- Львов
- Ровно
- Мариуполь
- Полтава
- Одесса
- Донецк
- Николаев
- Запорожье
- Луганск
- Харьков



порядке, если только управляющие флаги не предписывают досрочно прекратить выполнение. Работой службы или приложения управляет весь стек, а не один конкретный модуль. Необязательные аргументы используются для контроля работы модуля.

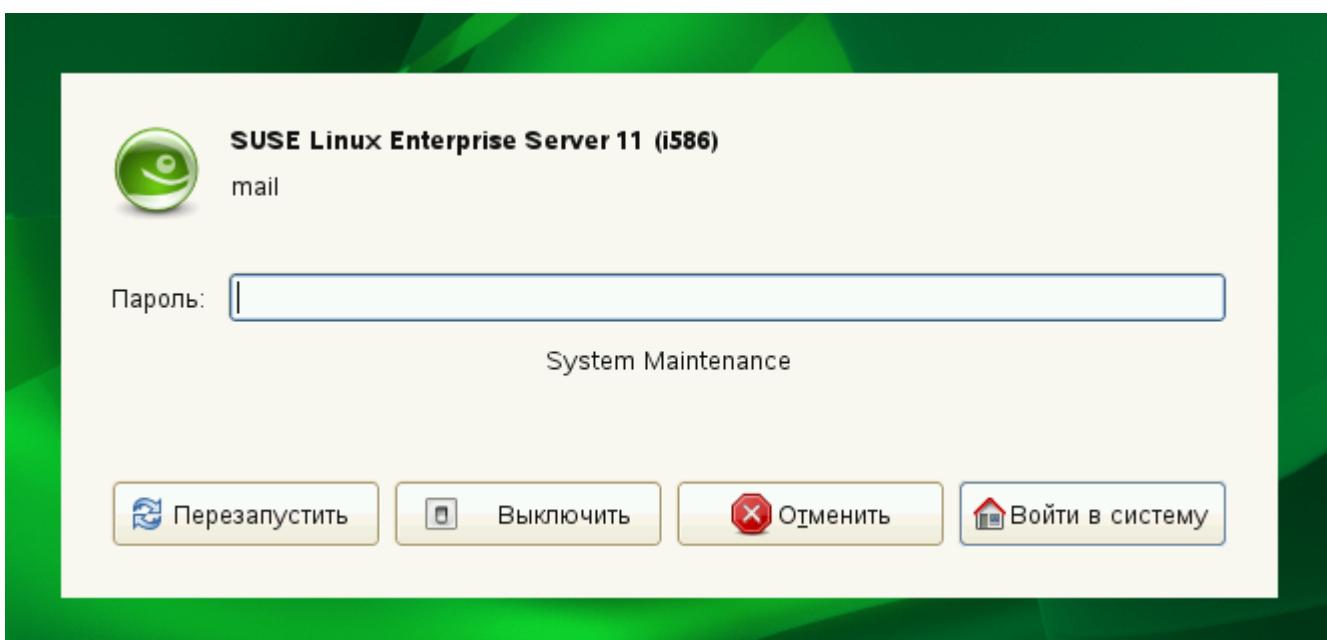
## 7.3 pam\_nologin.so

Данный модуль принимает тип *auth* и, по умолчанию, проверяет наличие файла */etc/nologin*. Если файл существует, то аутентификацию может пройти только пользователь *root*. Содержимое файла */etc/nologin* отображается при попытке входа пользователя в виде сообщения. Например, если мы проводим техническое обслуживание системы, и на время работ запретили вход непrivилегированным пользователям, то сообщение об этом пользователи увидят так:

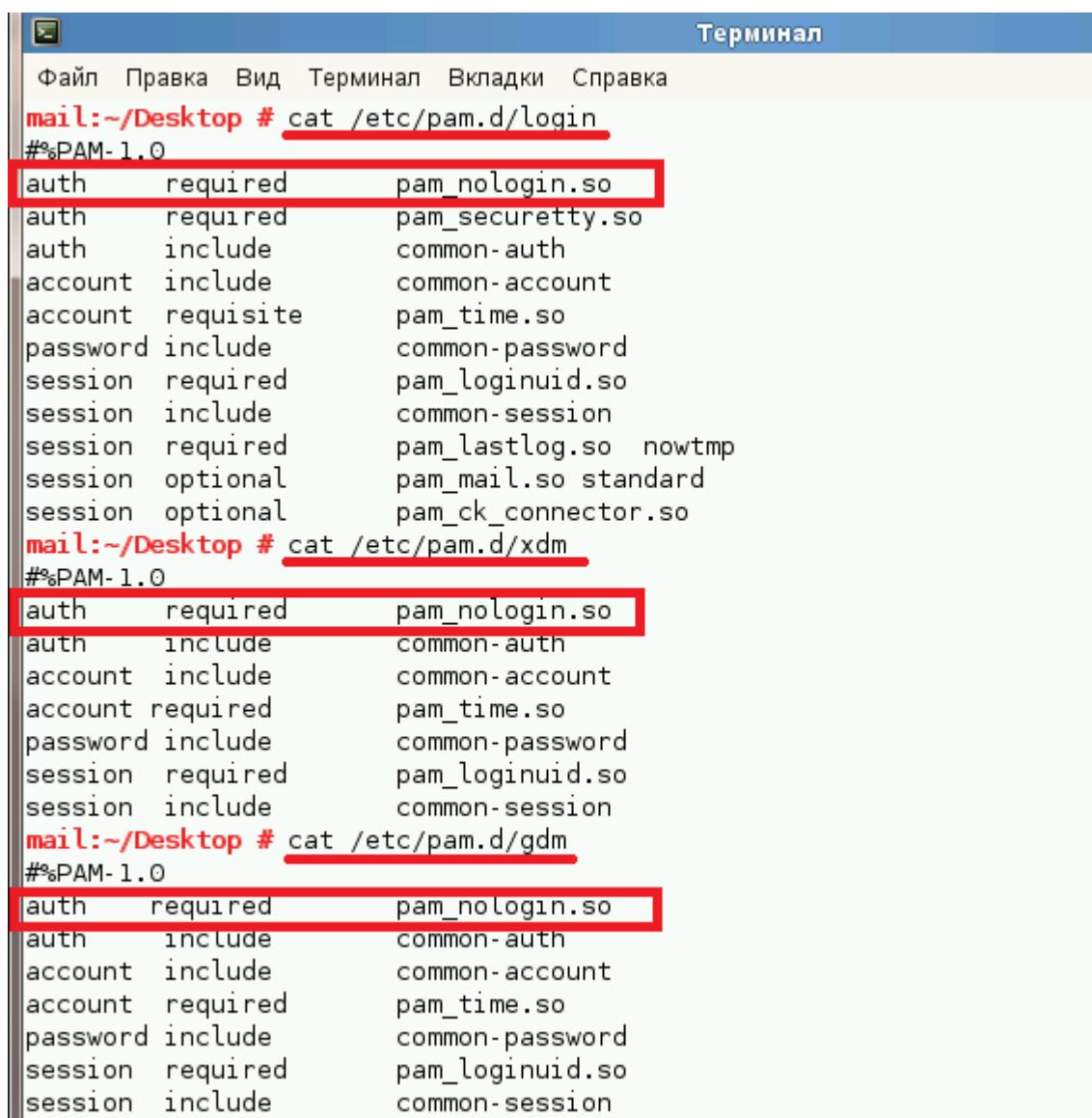
```
Welcome to SUSE Linux Enterprise Server 11 (i586)

mail login: oleg
System Maintenance

Имя пользователя неверно
```



Для такого эффекта необходимо создать файл `/etc/nologin` с текстом "System Maintenance" и настроить необходимые файлы из каталога `/etc/pam.d`. Это файл `/etc/pam.d/Login` (аутентификации в текстовом терминале), `/etc/pam.d/xdm` (аутентификация в графической оболочке), `/etc/pam.d/gdm` (аутентификация в GNOME), `/etc/pam.d/kdm` (аутентификация в KDE):



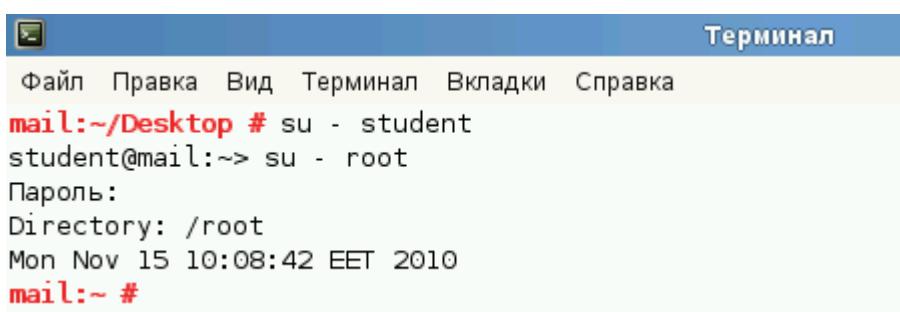
```
mail:~/Desktop # cat /etc/pam.d/login
 #%PAM-1.0
auth    required    pam_nologin.so
auth    required    pam_securetty.so
auth    include     common-auth
account include   common-account
account requisite pam_time.so
password include  common-password
session  required  pam_loginuid.so
session  include   common-session
session  required  pam_lastlog.so  nowtmp
session  optional   pam_mail.so standard
session  optional   pam_ck_connector.so
mail:~/Desktop # cat /etc/pam.d/xdm
 #%PAM-1.0
auth    required    pam_nologin.so
auth    include     common-auth
account include   common-account
account required  pam_time.so
password include  common-password
session  required  pam_loginuid.so
session  include   common-session
mail:~/Desktop # cat /etc/pam.d/gdm
 #%PAM-1.0
auth    required    pam_nologin.so
auth    include     common-auth
account include   common-account
account required  pam_time.so
password include  common-password
session  required  pam_loginuid.so
session  include   common-session
```

После действий, указанных выше, вы можете создавать или удалять файл `/etc/nologin`, соответственно запрещая или разрешая непrivилегированным пользователям вход в систему. Пользователь `root` будет иметь доступ в систему в любом случае, однако, содержимое файла `/etc/nologin` для него тоже будет отображаться при входе в систему.



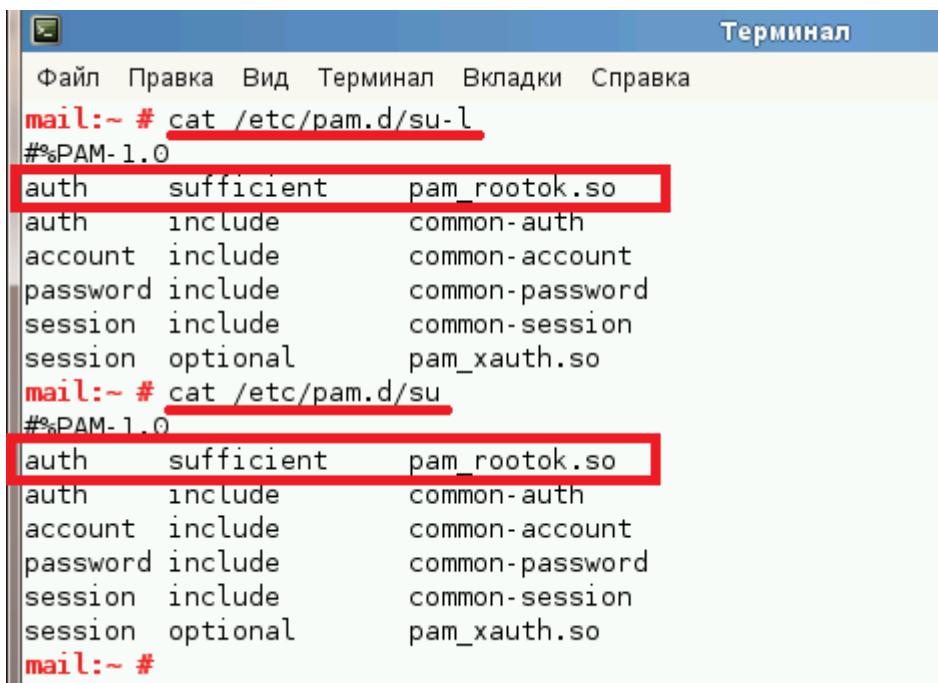
## 7.4 pam\_rootok.so

Данный модуль принимает тип *auth* и проверяет, является ли аутентифицирующийся пользователь пользователем *root*. Данный модуль, как правило, используется с флагом *sufficient*. Например, утилита *su* по умолчанию настроена таким образом: пользователи, которые хотят «превратиться» в других пользователей, должны знать пароль целевого пользователя. Однако, у пользователя *root* не запрашивается пароль целевого пользователя:



```
mail:~/Desktop # su - student
student@mail:~> su - root
Пароль:
Directory: /root
Mon Nov 15 10:08:42 EET 2010
mail:~ #
```

В каталоге */etc/pam.d* может существовать два файла для утилиты *su*. Первый - */etc/pam.d/su*, второй - */etc/pam.d/su-l*, который используется для утилиты *su* с аргументом “-1” или просто “-“.



```
mail:~ # cat /etc/pam.d/su-l
#%PAM-1.0
auth sufficient pam_rootok.so
auth include common-auth
account include common-account
password include common-password
session include common-session
session optional pam_xauth.so
mail:~ # cat /etc/pam.d/su
#%PAM-1.0
auth sufficient pam_rootok.so
auth include common-auth
account include common-account
password include common-password
session include common-session
session optional pam_xauth.so
mail:~ #
```





Работа стека *auth* в данном случае производится таким образом: пользователя первым делом пытаются «опознать», является ли он пользователем *root*. Если проверка проходит успешно – работа стека немедленно прерывается, и он признается успешно пройденным. Следовательно, модули, которые отвечают за запрос пароля, попросту не запускаются, что позволяет пользователю *root* избежать этого этапа проверки. Отключим использование модуля *pam\_rootok.so* и проверим эффект от данного действия:

Терминал

```
Файл Правка Вид Терминал Вкладки Справка
mail:~ # cat /etc/pam.d/su
#%PAM-1.0
#auth sufficient pam_rootok.so
auth include common-auth
account include common-account
password include common-password
session include common-session
session optional pam_xauth.so
mail:~ # cat /etc/pam.d/su-1
#%PAM-1.0
#auth sufficient pam_rootok.so
auth include common-auth
account include common-account
password include common-password
session include common-session
session optional pam_xauth.so
mail:~ # su student
Password:
su: incorrect password
mail:~ # su - student
Password:
su: incorrect password
mail:~ #
```

Итак, пользователь *root* не знает пароль пользователя *student* и не смог воспользоваться утилитой *su*. Такого эффекта мы добились, закомментировав в каждом из выше приведенных файлов строку *auth sufficient pam\_rootok.so*.

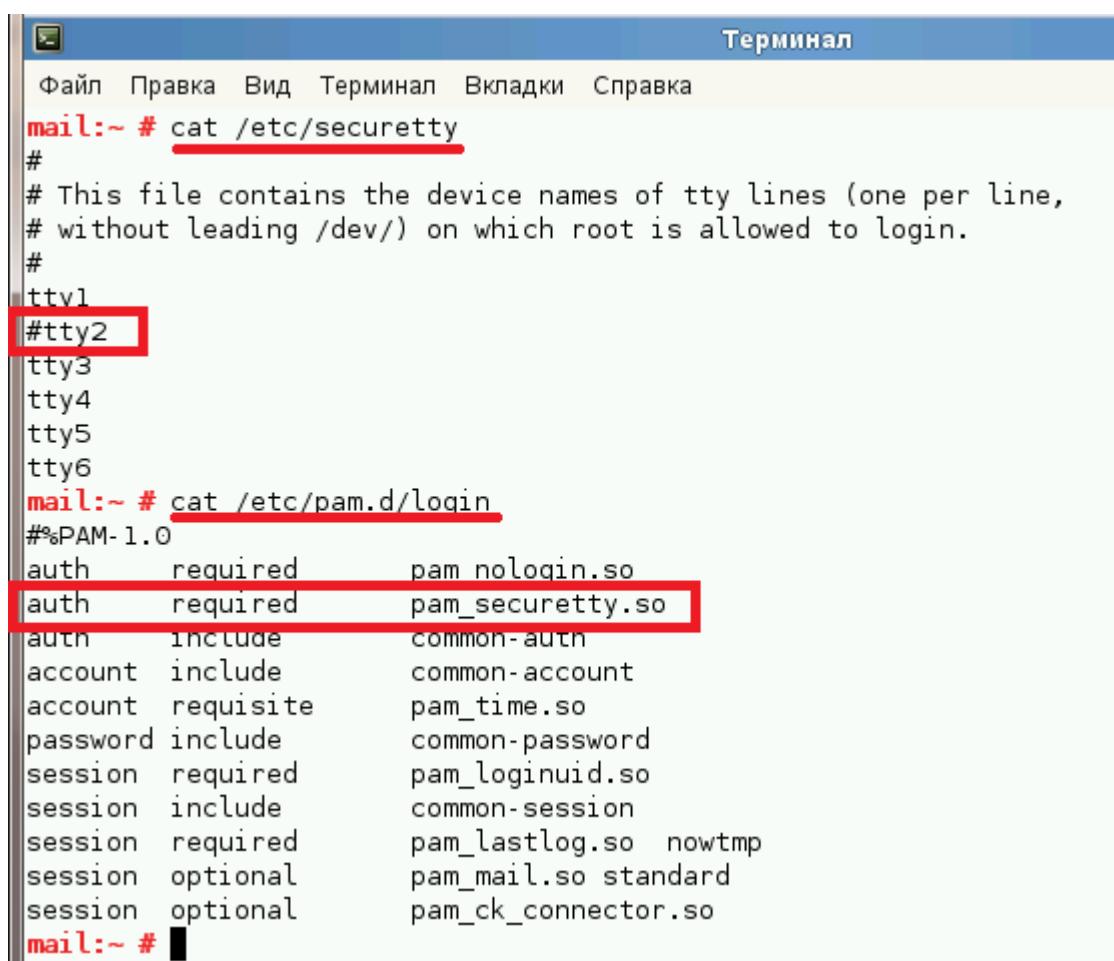
Безусловно, модулем *pam\_rootok.so* следует пользоваться очень осторожно. Например, если вы настроите утилиту *Login* аналогично утилите *su*, так, что пользователь *root* будет проходить аутентификацию без использования пароля, то это чревато тем, что каждый желающий будет работать в системе под учетной записью суперпользователя. Что, безусловно, недопустимо.

- Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава
- Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков

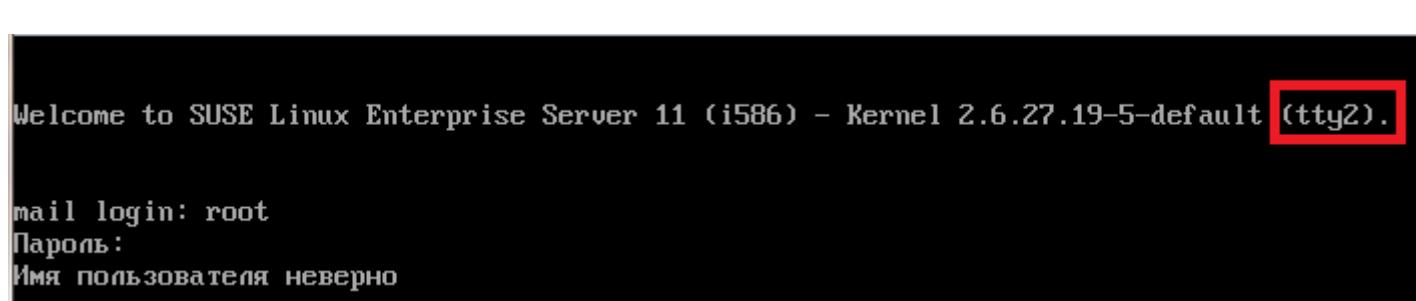


## 7.5 pam\_securetty.so

Данный модуль принимает тип *auth* и проверяет содержимое файла */etc/securetty*. В данном файле указывается список безопасных устройств, которые может использовать пользователь *root* для аутентификации в системе. Для примера, настроим данный модуль таким образом, чтобы пользователь *root* не мог выходить в систему со второго текстового терминала (*tty2*):



```
mail:~ # cat /etc/securetty
#
# This file contains the device names of tty lines (one per line,
# without leading /dev/) on which root is allowed to login.
#
ttv1
#tty2
tty3
tty4
tty5
tty6
mail:~ # cat /etc/pam.d/login
#%PAM-1.0
auth    required        pam_nologin.so
auth    required        pam_securetty.so
auth    include         common-auth
account  include        common-account
account  requisite      pam_time.so
password include       common-password
session   required      pam_loginuid.so
session   include        common-session
session   required      pam_lastlog.so  nowtmp
session   optional       pam_mail.so standard
session   optional       pam_ck_connector.so
mail:~ #
```



```
Welcome to SUSE Linux Enterprise Server 11 (i586) – Kernel 2.6.27.19-5-default (tty2)

mail login: root
Пароль:
Имя пользователя неверно
```



Обратите внимание на то, что модуль *pam\_securetty.so* не сообщает истинную причину неудачной аутентификации пользователя *root*. Это делается из дополнительных соображений безопасности. Действительно, если бы модуль сообщал о том, что устройство недопустимо для входа пользователь *root*, то это потенциальный злоумышленник просто сделал бы попытку входа с другого терминала. В нашем же случае, потенциальный злоумышленник, который предварительно взломал пароль, может сделать вывод, что пользователь *root* уже сменил свой пароль, и прекратить свои попытки доступа в систему.

## 7.6 pam\_time.so

С помощью данного модуля мы имеем возможность ограничить время работы пользователей в системе. Модуль *pam\_time.so* может иметь только тип *account*. Он не принимает никаких аргументов, но у него есть конфигурационный файл */etc/security/time.conf*. Назначение полей данного файла:

*службы ; терминалы ; пользователи ; время*

Поле	Описание
<i>Службы</i>	Логический список служб, допускается наличие нескольких записей для одной и той же службы
<i>Терминалы</i>	Логический список устройств доступа. Обычно консолям соответствуют имена <i>tty1</i> , <i>tty2</i> и т.д. Последовательным портам – <i>ttyS0</i> , <i>ttyS1</i> . Псевдотерминалам (сетевые соединения и сеансы X Window System) – <i>ttyp1</i> , <i>ttyp2</i> .
<i>Пользователи</i>	Логический список пользователей. Может включать пользователя <i>root</i> .
<i>Время</i>	Логический список дат, к которым применяется правило.

Под логическим списком подразумевается конструкция, элементы которой разделяются специальными операторами. Логические операторы можно смешивать, например *tty\* & !ttyp\** означает, что данным правилом разрешены любые последовательные устройства, но не псевдо терминалы.

Назначение логических операторов:

- Киев   ■ Днепропетровск   ■ Львов   ■ Ровно   ■ Мариуполь   ■ Полтава
- Одесса   ■ Донецк   ■ Николаев   ■ Запорожье   ■ Луганск   ■ Харьков





Оператор	Функция	Пример
&	Логическое умножение	пользователь1 & пользователь2 – правило применяется к обоим пользователям
/	Логическое сложение	tty1   tty2 – правило применяется либо к устройство tty1 либо к устройству tty2
!	Логическое отрицание	! login – правило не применяется к службе логин
*	Метасимвол	Соответствует любому значению. Интерпретация зависит от поля.

Коды дней с примерами:

Коды дней	Описание
Mo, Tu, We, Th, Fr, Sa, Su	Буквенные обозначения дня недели: понедельник, вторник и т.д. Допускается запись вида MoTuWe – что означает «понедельник вторник и среда».
Wk, Wd	Будние и выходные дни соответственно. Запись MoWk означает все будние дни кроме понедельника.
Al	Все дни недели. AlSu – все дни, кроме субботы.

Рассмотрим примеры:

\* ; tty1 ; root ; Al0000-2400

В данном примере суперпользователь имеет доступ ко всем службам в любое время, но только с терминала *tty1*.

Login | telnet; \* ; guest | dayuser ; Al0900-1800

Пользователи *guest* и *dayuser* имеют доступ в систему каждый день с 9.00 до 18.00 с любого терминального устройства, при условии, что подключение осуществляется через службу *Login* или *telnet*.

ftp ; \* ; \* ; Wk1800-2300

По будним дням (с понедельника по пятницу) любой пользователь может регистрироваться в нашей системе через *ftp* с 18:00 до 23:00.

ssh ; nightuser ; Al0000-0900&!Wd0400-0500

- Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава
- Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков

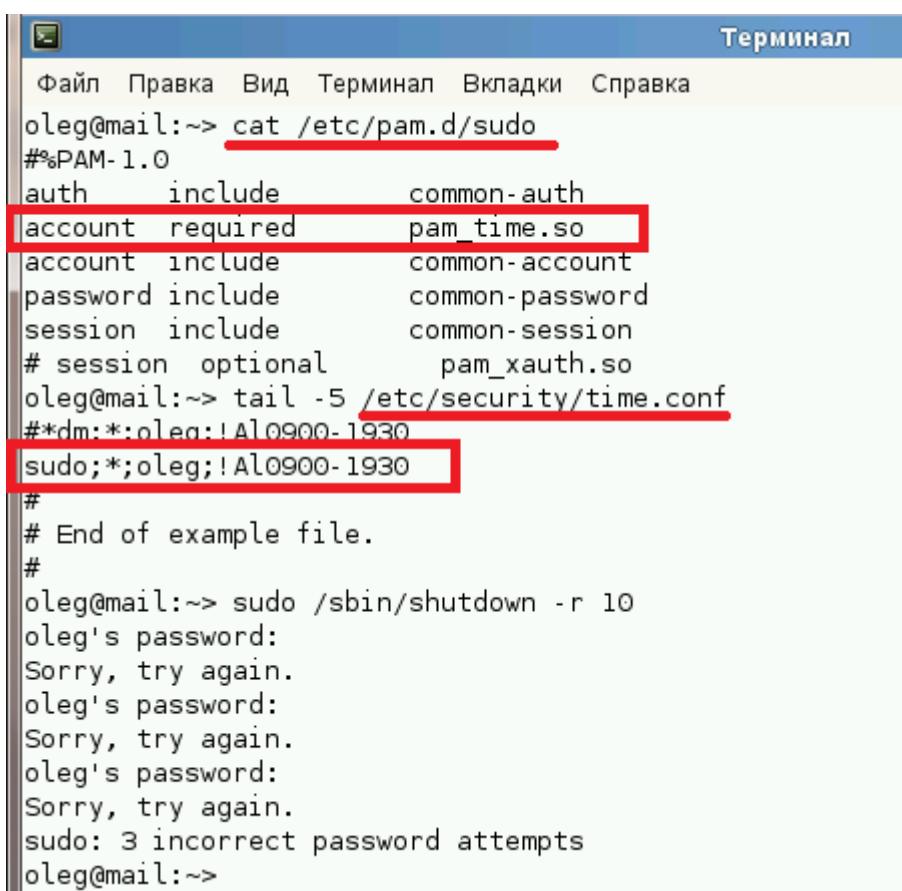


Пользователь *nightuser* имеет возможность осуществить вход в систему, используя службу *ssh*, в любой день с 0:00 до 9:00, за исключением периода с 4.00 до 5.00 в праздничные дни (суббота и воскресенье)

```
login ; * ; !student ; Wd0000-2400 | Wk1800-0800
```

Все пользователи, за исключением пользователя *student*, могут входить в систему только в нерабочее время.

После того, как необходимые параметры файла */etc/security/time.conf* настроены, можно подключать модуль *pam\_time.so* в соответствующий стек. Например:



Терминал

```

Файл Правка Вид Терминал Вкладки Справка
oleg@mail:~> cat /etc/pam.d/sudo
#%PAM-1.0
auth    include      common-auth
account required    pam_time.so
account include      common-account
password include     common-password
session include     common-session
# session optional   pam_xauth.so
oleg@mail:~> tail -5 /etc/security/time.conf
##dm:*:oleg:!Al0900-1930
|sudo;*:oleg;!Al0900-1930
#
# End of example file.
#
oleg@mail:~> sudo /sbin/shutdown -r 10
oleg's password:
Sorry, try again.
oleg's password:
Sorry, try again.
oleg's password:
Sorry, try again.
sudo: 3 incorrect password attempts
oleg@mail:~>
```

На примере пользователю *oleg* запретили пользоваться утилитой *sudo* с 9.00 до 19.30 каждого дня. При попытке аутентификации пользователю сообщается, что он вводит неправильный пароль. В очередной раз мы сталкиваемся с ситуацией, когда модуль не сообщает об истинной причине неудачной аутентификации. Такое поведение является характерной чертой большинства модулей. Чем меньше информации предоставляется для «нарушителей», тем более безопасна система.



## 7.7 pam\_limits.so

Ограничить возможности пользователей возможно не только по времени доступа, но и лимитируя доступ к системным ресурсам в каждом сеансе. Это полезно для защиты системы от атак типа "отказ от обслуживания".

Модуль *pam\_limits.so* может иметь только тип *session*. Поддерживает он два аргумента: *debug* и *conf=конфигурационный\_файл* (по умолчанию *- /etc/security/limits.conf*). На суперпользователя никакие ограничения не действуют.

В файле */etc/security/limits.conf* задаются ограничения для отдельных пользователей и для групп. Все ограничения действуют в рамках одного сеанса. Каждая строка представляет собой отдельную запись. Синтаксис таков:

*Пользователь или @группа; тип ; ресурс ; лимит*

Поле	Описание
<i>Пользователь или @группа</i>	Используется имя пользователя или имя группы, с предшествующим знаком «@». Допускается использование метасимвола «*», означающего «все пользователи».
<i>Тип (hard или soft)</i>	Поле содержит тип ограничения. hard – фиксированный лимит. soft – лимит, который может быть превышен, но информация об этом отобразится в журнальных файлах.
<i>Ресурс</i>	<p><i>fsize</i> – максимальный размер используемых файлов (КБайт)</p> <p><i>nofile</i> – максимальное количество одновременно открытых файлов</p> <p><i>procs</i> – максимальное число запущенных процессов</p> <p><i>maxlogins</i> – максимальное число одновременных сеансов работы пользователя</p>
<i>Лимит</i>	Собственно лимит на указанный ресурс. Как правило, задается в числовом виде.

Ограничения применяются только в рамках одного действующего сеанса работы. С помощью ресурса *maxLogins* можно ограничить количество таких сеансов работы. По умолчанию, как правило, модуль *pam\_limits.so* указан в стеке файла */etc/pam.d/common-session*. Содержимое файлов */etc/pam.d/common-\** включается во многие файлы каталога */etc/pam.d* посредством опции *include*:

- Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава
- Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков





```
Терминал
Файл Правка Вид Терминал Вкладки Справка
mail:~/Desktop # cat /etc/pam.d/login
#%PAM-1.0
auth    required      pam_nologin.so
auth    required      pam_securetty.so
auth    include       common-auth
account include     common-account
account requisite   pam_time.so
password include   common-password
session required    pam_loginuid.so
session include     common-session
session required    pam_lastlog.so  nowtmp
session optional   pam_mail.so standard
session optional   pam_ck_connector.so
mail:~/Desktop # cat /etc/pam.d/common-session
#%PAM-1.0
#
# This file is autogenerated by pam-config. All changes
# will be overwritten.
#
# Session-related modules common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of modules that define tasks to be performed
# at the start and end of sessions of *any* kind (both interactive and
# non-interactive
#
session required    pam_limits.so
session required    pam_unix2.so
session optional   pam_umask.so
mail:~/Desktop #
```

Вот такое сообщение получает пользователь, в случае, если он превысил количество допустимых сеансов работы:

```
Welcome to SUSE Linux Enterprise Server 11 (i586) – Kernel 2.6.27.19-5-default (tty4).

mail login: oleg
Пароль:
Слишком много регистраций в системе для 'oleg'.
Последний вход в систему: Пнд Ноя 15 20:36:34 EET 2010 на tty4

Доступ запрещен
```



А на следующем изображении показано, что пользователь *oleg* уже находится в системе, войдя в нее со второго терминала. И что для него задано ограничение максимального числа одновременных сеансов работы:

**Терминал**

```

Файл Правка Вид Терминал Вкладки Справка
mail:~/Desktop # tail /etc/security/limits.conf
#*           hard    rss      10000
#@student    hard    nproc     20
#@faculty    soft    nproc     20
#@faculty    hard    nproc     50
#ftp         hard    nproc      0
#@student    -       maxlogins  4
oleg        -       maxlogins  1
#oleg       hard    nproc     20

# End of file
mail:~/Desktop # who
root    ttys1          2010-11-15 09:50
oleg    tty2          2010-11-15 20:36
root    ttys2          2010-11-15 09:51
root    :0            2010-11-15 09:51 (console)
root    pts/0          2010-11-15 20:37 (:0.0)
mail:~/Desktop #

```

## 7.8 Задание для самопроверки

1. Создайте гостевую учетную запись. Ограничите количество запускаемых процессов - количеством 10, количество одновременных регистраций в терминалах - 2. Максимальный размер файла, созданного данным пользователем - 100МБ. Доступные для регистрации терминалы - от 1-го до 3-го.
2. Ограничите круг пользователей, которым разрешено использование утилиты *sudo* группой *wheel*.
3. Что будет происходить, если пользователь запустит сценарий, вызывающий самого себя? Что можно предпринять, чтобы защитить систему от такой ситуации?
4. Что будет происходить, если пользователь запустит сценарий, создающий каталог, затем изменяющий текущее местоположение на созданный каталог, и вызывающий после этого самого себя? Как с этим бороться?



## 8. Управление процессами

Процесс - это абстракция, используемая в Linux для описания выполняющейся программы. Процесс представляет собой системный объект, посредством которого можно контролировать обращения программы к памяти, центральному процессору и ресурсам ввода-вывода.

Концепция, согласно которой как можно больше работы должно выполняться в контексте процессов, а не в ядре, является частью философии UNIX и Linux. Системные и пользовательские процессы подчиняются одним и тем же правилам, благодаря чему управление ими осуществляется с помощью единого набора команд.

### 8.1 Атрибуты процесса

Процесс состоит из адресного пространства и набора структур данных, содержащихся внутри ядра. Адресное пространство представляет собой совокупность страниц памяти, которые были внесены ядром для выполнения процесса. В него загружается код процесса и используемые им библиотеки функций, а также переменные, содержимое стеков и различная вспомогательная информация, необходимая ядру во время работы процесса. поскольку в Linux поддерживается концепция виртуальной памяти, страницы адресного пространства в конкретный момент времени могут находиться либо в физической памяти, либо в разделе подкачки, т.е. на диске.

В структурах данных ядра хранится всевозможная информация о каждом процессе. К наиболее важным сведениям относятся:

- таблица распределения памяти;
- текущий статус (неактивен, приостановлен, выполняется и т.п.);
- приоритет
- информация об используемых ресурсах;
- маска сигналов (запись о том, какие сигналы блокируются);
- идентификатор владельца.

Многие характеристики процесса непосредственно влияют на его выполнение. В частности, имеет значение, сколько времени выделяется ему центральным процессором, к каким файлам он имеет доступ и т.д. Соберем воедино самые



интересные с точки зрения системного администратора характеристики процессов. Они поддерживаются во всех версиях UNIX и Linux.

### **8.1.1 Идентификатор процесса (PID)**

Каждому процессу назначается уникальный идентификатор (Process ID, PID). Большинство команд и системных вызовов, работающих с процессами, требует указания конкретного идентификатора, чтобы был ясен контекст операции. Идентификаторы присваиваются по порядку по мере создания процессов. Когда номера заканчиваются, ядро сбрасывает значение счетчика в единицу и снова начинает присваивать их по порядку, пропуская те идентификаторы, которые еще используются.

### **8.1.2 Идентификатор родительского процесса (PPID)**

В Linux нет системного вызова, который создавал бы новый процесс для выполнения конкретной программы. Существующий процесс должен клонировать сам себя, чтобы породить новый процесс. Клон может заменить выполняемую программу другой.

В операции клонирования исходный процесс называют родительским, а его клон - дочерним. Помимо собственного идентификатора каждый дочерний процесс имеет атрибут PPID (Parent Process ID), который совпадает с идентификатором породившего его родительского процесса.

### **8.1.3 Идентификатор пользователя (UID) и эффективный идентификатор пользователя (EUID).**

В UID (User ID) - это идентификатор пользователя, создавшего данный процесс, точнее, копия EUID родительского процесса. Менять атрибуты процесса могут только его создатель (владелец) и root.

EUID (Effective User ID) — это "эффективный" пользовательский идентификатор процесса, предназначенный для того, чтобы определить, к каким ресурсам и файлам у процесса есть право доступа в данный момент. У большинства процессов значения UID и EUID одинаковы. Исключение составляют программы, у которых установлен бит смены идентификатора пользователя (SUID).

Зачем нужны два идентификатора? Просто имеет смысл разграничить понятия персонификации и прав доступа. К тому же программы с установленным битом SUID не всегда выполняются с расширенными привилегиями. Значение EUID можно устанавливать и сбрасывать, чтобы предоставлять процессу дополнительные полномочия или отбирать их.

Операционная система Linux хранит начальный идентификатор, т.е. копию значения EUID, которое имел процесс в начальной точке. Если процесс не удалит сохраненный идентификатор, его можно будет использовать в качестве реального или эффективного идентификатора. Благодаря этому "консервативно" написанная программа с установленным битом SUID может большую часть времени работать с обычными привилегиями, переходя в режим расширенных привилегий лишь в определенные моменты.

#### **8.1.4 Идентификатор группы (GID) и эффективный идентификатор группы (EGID)**

Эффективный идентификатор группы (Effective Group ID, EGID) связан с атрибутом GID так же, как значение EUID — с UID, т.е. они будут отличаться в случае программы, у которой установлен бит смены идентификатора группы (SGID). Начальное значение GID тоже сохраняется.

В Linux процесс одновременно может относиться к нескольким группам. Список групп хранится отдельно от значений GID и EGID. При анализе прав доступа проверяется эффективный идентификатор и дополнительный список групп, но не значение GID.

Если пользователь пытается получить доступ к какому-либо ресурсу, весь список перепроверяется на предмет того, принадлежит ли пользователь к группе, членам которой разрешается использовать данный ресурс.

#### **8.1.5 Приоритет и фактор уступчивости**

Приоритет процесса определяет долю времени центрального процессора, которую получает программа. Ядро применяет динамический алгоритм вычисления приоритетов, учитывающий, сколько времени центрального процессора уже

- Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава
- Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков



использовал процесс и сколько времени он ожидает своей очереди. Кроме того, учитывается заданный администратором, так называемый, фактор уступчивости (устанавливается с помощью программы nice), который определяет, в какой степени программа может "делиться" процессором с другими программами.

### 8.1.6 Управляющий терминал

С большинством процессов связан управляющий терминал, который определяет базовую конфигурацию стандартных каналов ввода, вывода и ошибок. Когда пользователь запускает какую-либо программу в интерпретаторе, его терминал, как правило, становится управляющим терминалом процесса. От управляющего терминала зависит также распределение сигналов.

## 8.2 Просмотр информации о процессах

Подробную информацию о запущенных процессах можно получить с помощью утилит *ps* и *top*. Утилита *ps* может делать только «разовый» снимок процессов. Если нужно постоянно следить за процессами, лучше использовать утилиту *top*.

### 8.2.1 ps

Утилита *ps*, в зависимости от переданных ей аргументов, выводит разную по наполнению информацию. Наиболее обширный список запущенных процессов можно получить, используя ключи *aux*.

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.0	1008	360	?	Ss	17:44	0:01	init [5]
root	2	0.0	0.0	0	0	?	S<	17:44	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	S<	17:44	0:00	[migration/0]
root	4	0.0	0.0	0	0	?	S<	17:44	0:00	[ksoftirqd/0]
root	5	0.0	0.0	0	0	?	S<	17:44	0:02	[events/0]

На изображении представлены лишь несколько строк вывода данной команды, обычно количество запущенных процессов настолько велико, что их список не помещается в рамках одного экрана вывода. Обратим свое внимание на заголовки полей, в которых отображается информация о процессах.





Поле	Описание
<i>USER</i>	Имя владельца процесса
<i>PID</i>	Идентификатор процесса
<i>%CPU</i>	Доля времени центрального процессора, выделенная процессу (в процентах)
<i>%MEM</i>	Часть реальной памяти, используемой процессом (в процентах)
<i>VSZ</i>	Виртуальный размер процесса
<i>RSS</i>	Размер резидентного набора страниц (количество страниц памяти)
<i>TTY</i>	Терминал, на котором запущен процесс
<i>STAT</i>	Текущее состояние процесса:  R – выполняется  D – ожидает записи на диск  S – неактивен (выполняется более 20 секунд)  T – приостановлен  Z – зомби  < – процесс с повышенным приоритетом (пониженным фактором уступчивости)  N – процесс с пониженным приоритетом (повышенным фактором уступчивости)  + – процесс выполняется не в фоновом режиме  I – многопоточное приложение
<i>START</i>	Время запуска процесса
<i>TIME</i>	Количество процессорного времени, затраченного на выполнение процесса
<i>COMMAND</i>	Имя и аргумента выполняющейся команды

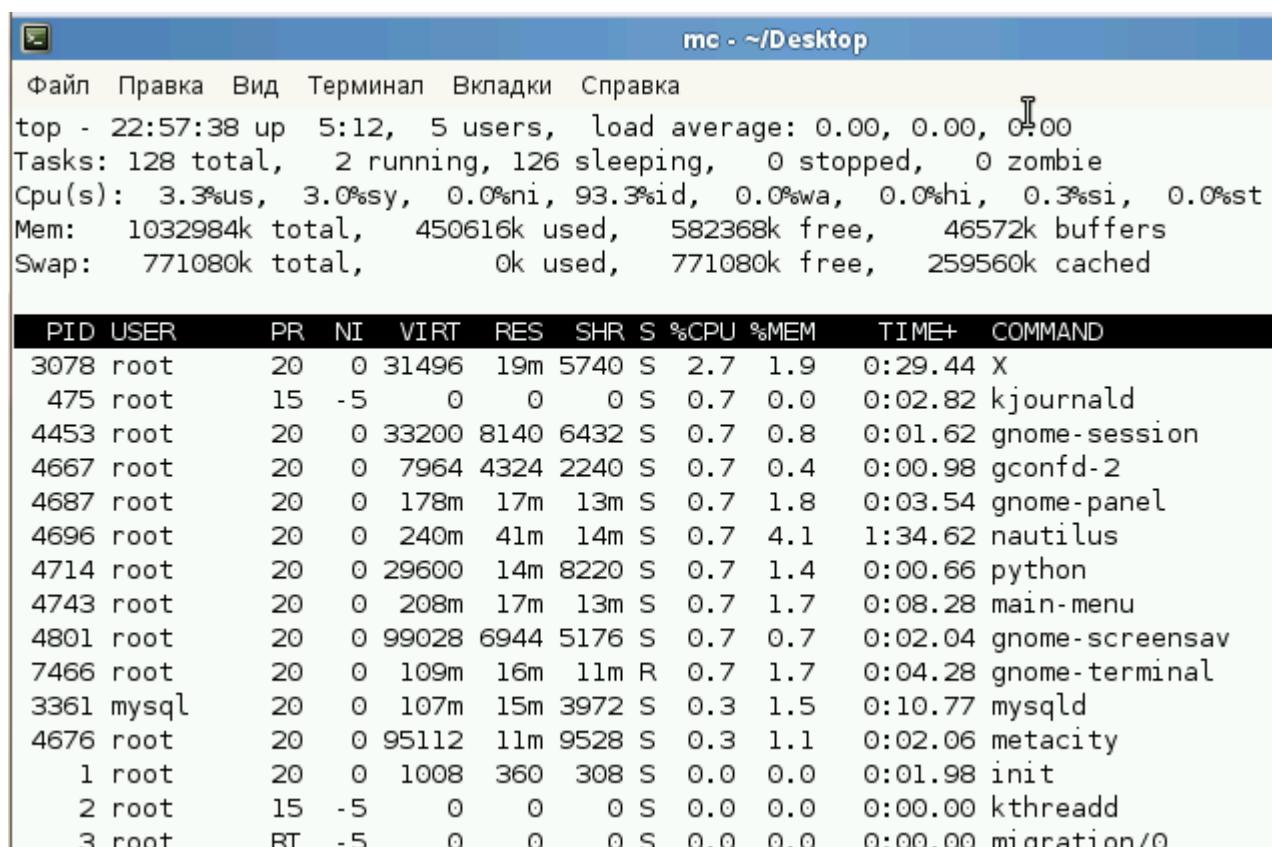




## 8.2.2 top

По умолчанию информация, которая выводится на экран командой *top*, обновляется каждые 10 секунд. Наиболее активные процессы отображаются первыми. Также, используя данную утилиту, можно передавать сигналы процессам и изменять их фактор уступчивости. Поля, которые отображаются в выводе данной команды, сходны с полями, выводимыми командой *ps*. Однако, команда *top* представляет еще и сводную статистику по процессам, что может быть чрезвычайно полезным при наблюдении за системой.

Для прокрутки списка процессов необходимо пользоваться знаками «» и «<».



```

mc - ~/Desktop
Файл Правка Вид Терминал Вкладки Справка
top - 22:57:38 up 5:12, 5 users, load average: 0.00, 0.00, 0.00
Tasks: 128 total, 2 running, 126 sleeping, 0 stopped, 0 zombie
Cpu(s): 3.3%us, 3.0%sy, 0.0%ni, 93.3%id, 0.0%wa, 0.0%hi, 0.3%si, 0.0%st
Mem: 1032984k total, 450616k used, 582368k free, 46572k buffers
Swap: 771080k total, 0k used, 771080k free, 259560k cached

PID USER      PR  NI  VIRT   RES   SHR S %CPU %MEM     TIME+ COMMAND
3078 root      20   0 31496 19m 5740 S  2.7  1.9  0:29.44 X
 475 root      15  -5    0    0   0 S  0.7  0.0  0:02.82 kjournald
4453 root      20   0 33200 8140 6432 S  0.7  0.8  0:01.62 gnome-session
4667 root      20   0 7964 4324 2240 S  0.7  0.4  0:00.98 gconfd-2
4687 root      20   0 178m 17m 13m S  0.7  1.8  0:03.54 gnome-panel
4696 root      20   0 240m 41m 14m S  0.7  4.1  1:34.62 nautilus
4714 root      20   0 29600 14m 8220 S  0.7  1.4  0:00.66 python
4743 root      20   0 208m 17m 13m S  0.7  1.7  0:08.28 main-menu
4801 root      20   0 99028 6944 5176 S  0.7  0.7  0:02.04 gnome-screensav
7466 root      20   0 109m 16m 11m R  0.7  1.7  0:04.28 gnome-terminal
3361 mysql     20   0 107m 15m 3972 S  0.3  1.5  0:10.77 mysqld
4676 root      20   0 95112 11m 9528 S  0.3  1.1  0:02.06 metacity
  1 root      20   0 1008 360 308 S  0.0  0.0  0:01.98 init
  2 root      15  -5    0    0   0 S  0.0  0.0  0:00.00 kthreadd
  3 root      RT  -5    0    0   0 S  0.0  0.0  0:00.00 migration/0

```

Во время работы утилиты *top* можно пользоваться встроенными в нее командами. Список команд можно получить, используя команду «*h*». Наиболее полезные команды – «*k*» – с помощью нее можно послать сигналы процессам, и «*r*» – с помощью нее можно изменять приоритет процесса.



## 8.3 Сигналы

Сигналы - это запросы на прерывание, реализуемые на уровне процессов. Определено свыше тридцати различных сигналов, и они находят самое разное применение.

- Сигналы могут посыпаться между процессами как средство коммуникации.
- Сигналы могут посыпаться драйвером терминала для уничтожения или приостановки процесса, когда пользователь нажимает специальные сочетания клавиш, такие как `<Ctrl+C>` или `<Ctrl+Z>`
- Сигналы могут посыпаться в самых разных целях пользователем или администратором с помощью команд `kill` и `killall`.
- Сигналы могут посыпаться ядром, когда процесс выполняет нелегальную инструкцию, например деление на ноль.

Когда поступает сигнал, возможен один из двух вариантов развития событий. Если процесс назначил сигналу подпрограмму обработки, то после вызова предоставляется информация о контексте, в котором был сгенерирован сигнал. В противном случае ядро выполняет от имени процесса действия, заданные по умолчанию. Эти действия зависят от сигнала. Многие сигналы приводят к завершению процесса, а в некоторых случаях при этом еще и создается дамп памяти (файл, содержащий образ памяти процесса, который можно использовать для отладки).

Процедура вызова обработчика называется перехватом сигнала. Когда выполнение обработчика завершается, процесс возобновляется с той точки, где был получен сигнал.

Чтобы определенные сигналы не поступали в программу, нужно задать их игнорирование или блокирование. Игнорируемый сигнал просто пропускается и не влияет на работу процесса. Блокируемый сигнал ставится в очередь на обработку, но ядро не требует от процесса никаких действий до явного разблокирования сигнала. Обработчик вызывается для разблокированного сигнала только один раз, даже если в течение периода блокировки поступило несколько аналогичных сигналов.

Перечислим сигналы, которые должны быть известны каждому системному администратору. Имена сигналов традиционно записываются прописными буквами. Иногда к именам добавляется префикс `SIG_`:



Номер сигнала	Имя	Описание	Реакция по умолчанию	Перехватывается?	Блокируется?	Дамп памяти?
1	<i>HUP</i>	Отбой	Завершение	ДА	ДА	НЕТ
2	<i>INT</i>	Прерывание	Завершение	ДА	ДА	НЕТ
3	<i>QUIT</i>	Выход	Завершение	ДА	ДА	ДА
9	<i>KILL</i>	Уничтожение	Завершение	НЕТ	НЕТ	НЕТ
15	<i>TERM</i>	Запрос на завершение	Завершение	ДА	ДА	НЕТ

### 8.3.1 Отправка сигналов командами *kill* и *killall*

Команду *kill* чаще всего используют для уничтожения процессов. Эта команда может послать процессу любой сигнал, но по умолчанию это сигнал TERM. Команду *kill* могут выполнять как рядовые пользователи (для своих собственных процессов), так и пользователь *root* (для любого процесса). Она имеет следующий синтаксис:

*kill -идентификатор\_сигнала PID*

Идентификатор сигнала - номер или символическое имя посылаемого сигнала, PID - номер искомого процесса.

Команда *kill* без номера не гарантирует завершение процесса, так как генерируемый по умолчанию сигнал TERM можно перехватывать, блокировать и игнорировать. Указав в качестве сигнала "-9" - мы практически гарантированно завершим процесс.

Если идентификатор процесса, которому нужно послать сигнал, неизвестен, необходимо воспользоваться командой *ps*. Другой вариант - команда *killall*, самостоятельно выполняющая такой поиск. Нужно лишь задать в аргументах команды *killall* имя процесса. При этом, если в системе запущено несколько одноименных процессов, всем им будет послан указанный сигнал.



## 8.4 Управление режимом работы процесса

По умолчанию, приложение, вызываемое в командном интерпретаторе, запускается в экранном режиме, «захватывая» таким образом терминал, и не позволяя вводить какие-либо команды до завершения своей работы. Запуск задания в фоновом режиме работы позволяет освободить терминал для других целей. Чтобы выполнить команду в фоновом режиме достаточно указать после нее специальный оператор «&».

Если же команда уже выполняется в экранном режиме, мы можем приостановить ее выполнение с помощью комбинации клавиш **<Ctrl>+<Z>**. После этого приложению присваивается номер задания, который отображается командой *jobs*. Для перевода задания в фоновый режим необходимо выполнить команду *bg N*, где *N* – номер задания. Для перевода задания в экранный режим, используется команда *fg N*.



```
Файл Правка Вид Терминал Вкладки Справка
mail:~/Desktop # updatedb
^Z
[1]+ Stopped                  updatedb
mail:~/Desktop # jobs
[1]+ Stopped                  updatedb
mail:~/Desktop # bg 1
[1]+ updatedb &
mail:~/Desktop # jobs
[1]+ Running                  updatedb &
mail:~/Desktop # fg 1
updatedb
```

Обратите внимание на то, что команда *jobs* отображает информацию о том, выполняется ли приложение в фоновом режиме, или его работа приостановлена

## 8.5 Задание для самопроверки

1. Запустите в экранном режиме с помощью команды *wget* скачивание какого-либо файла из сети Internet. Переведите этот процесс в фоновый режим работы. Выделите ему больше процессорного времени, изменив приоритет.
2. Запустите пользователем *student* несколько окон программы *xcalc*. Остановите работу всех экземпляров данной программы одной командой под учетной записью пользователя *root*.



## 9. Основы shell-программирования

Применение shell-сценариев позволяет экономно расходовать рабочее время при выполнении важных и сложных заданий. В этих сценариях используются переменные, условные, арифметические и циклические конструкции, которые можно отнести к системным командам. За счет этих возможностей сценарии создаются довольно быстро. Интерпретатор команд может применять в качестве вводных данных для одной команды информацию, полученную при выполнении другой команды. Чтобы shell-сценарий применялся в различных системах UNIX и Linux, в него нужно внести лишь небольшие изменения. Это связано с тем, что интерпретаторы обладают высокой степенью универсальности. Определенные трудности возникают лишь вследствие ориентации системных команд на определенные системы

### 9.1 Структура сценария

Сценарий не относится к компилируемым программам, поскольку он интерпретируется построчно. Первой строкой сценария всегда должна быть строка вида: `#!/bin/bash`. Каждый сценарий может содержать комментарии; если комментарии помещаются в сценарии, первым символом в строке комментария будет символ `#`. Встретив подобный символ, интерпретатор команд игнорирует строку комментария.

Сценарий просматривается интерпретатором команд в направлении сверху вниз. Перед выполнением сценария требуется воспользоваться командой `chmod`, предоставляющей право доступа на выполнение. Убедитесь в том, что в переменных окружения правильно указаны пути к сценариям, тогда для их выполнения достаточно будет указать имя файла сценария.

### 9.2 Проверка условий

Проверка условия – одно из важнейших действий в сценарии. Логика работы сценария зачастую зависит от выполнения какого-либо события, наличия какого-либо файла или определенного символа, введенного пользователем с клавиатуры.

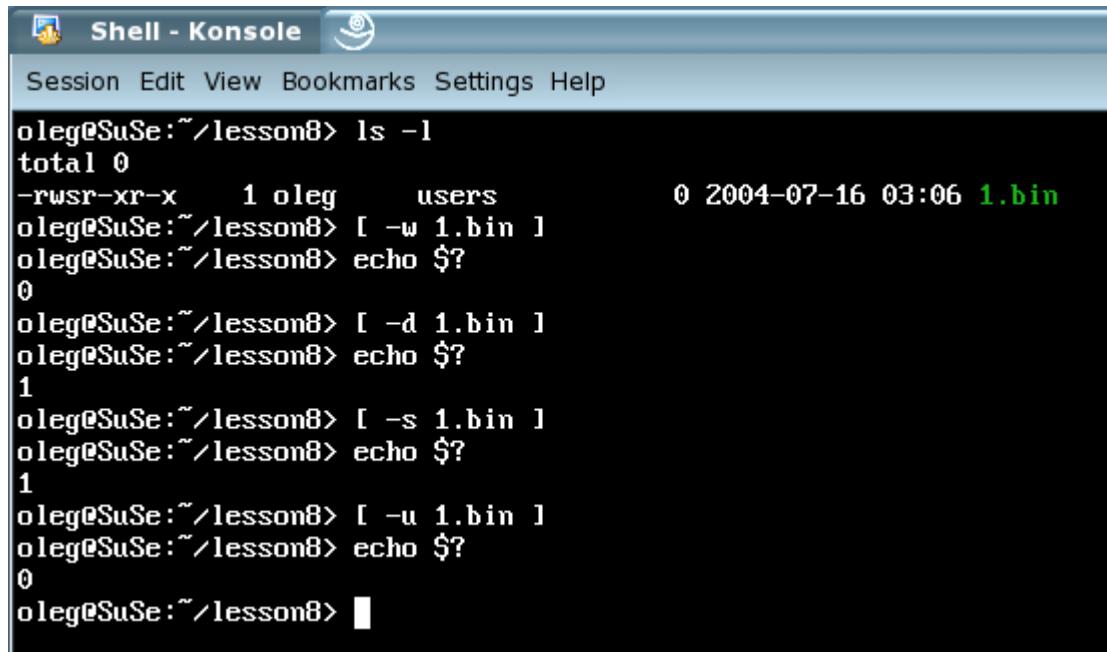
#### 9.2.1 Проверка прав доступа к файлу

- Киев
- Днепропетровск
- Львов
- Ровно
- Мариуполь
- Полтава
- Одесса
- Донецк
- Николаев
- Запорожье
- Луганск
- Харьков



При проверке права доступа к файлу может применяться большое количество условий:

Условие	Объяснение
-d	Каталог
-f	Обычный файл
-L	Символическая связь
-r	Файл доступен для чтения
-s	Файл имеет ненулевой размер, он не пуст
-w	Файл доступен для записи
-u	Файл имеет установленный бит SUID
-x	Файл доступен для запуска



```

Shell - Konsole

Session Edit View Bookmarks Settings Help

oleg@SuSe:~/lesson8> ls -l
total 0
oleg@SuSe:~/lesson8> [ -w 1.bin ]
oleg@SuSe:~/lesson8> echo $?
0
oleg@SuSe:~/lesson8> [ -d 1.bin ]
oleg@SuSe:~/lesson8> echo $?
1
oleg@SuSe:~/lesson8> [ -s 1.bin ]
oleg@SuSe:~/lesson8> echo $?
1
oleg@SuSe:~/lesson8> [ -u 1.bin ]
oleg@SuSe:~/lesson8> echo $?
0
oleg@SuSe:~/lesson8>

```

Обратите внимание на то, что результат проверки условия мы получаем, обращаясь к переменной `$?`, которая хранит в себе результат выполнения последней команды. Значение `0`, принимаемое данной переменной, означает, что условие выполнено, значение `1` - если нет.



## 9.2.2 Применение логических операторов

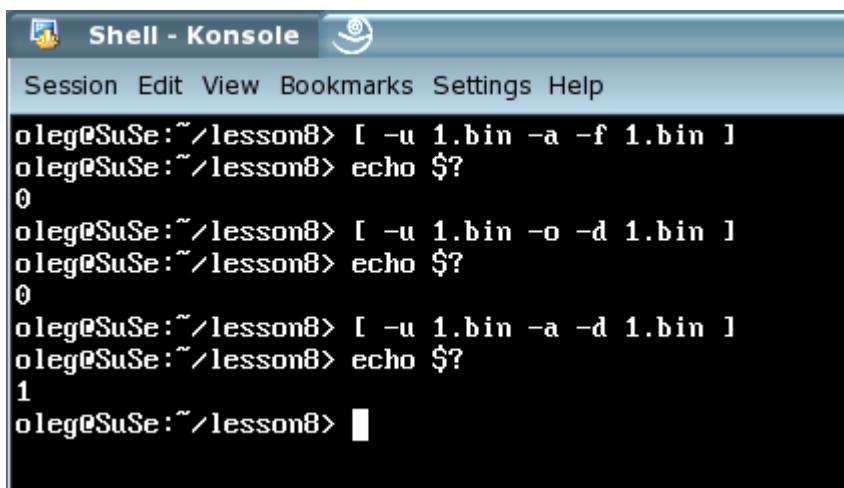
Иногда возникает необходимость в сравнении различных прав доступа. Чтобы реализовать подобную проверку, существуют логические операторы:

---

**Условие    Объяснение**

---

- a*      Логическое AND, возвращает истину, если обе части оператора принимают истинное значение
  - o*      Логическое OR, возвращает истину, если любая из частей оператора принимает истинное значение
- 



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
oleg@SuSe:~/lesson8> [ -u 1.bin -a -f 1.bin ]
oleg@SuSe:~/lesson8> echo $?
0
oleg@SuSe:~/lesson8> [ -u 1.bin -o -d 1.bin ]
oleg@SuSe:~/lesson8> echo $?
0
oleg@SuSe:~/lesson8> [ -u 1.bin -a -d 1.bin ]
oleg@SuSe:~/lesson8> echo $?
1
oleg@SuSe:~/lesson8> 
```

## 9.2.3 Проверка строк

Проверка строк является важным этапом при отслеживании ошибок. Чтобы проверить строки достаточно воспользоваться одним из форматов:

```
test строка
test оператор_строки строка
test "строка" оператор_строки "строка"

[ оператор_строки строка ]
[ строка оператор_строки строка ]
```

Операторами строк могут быть:

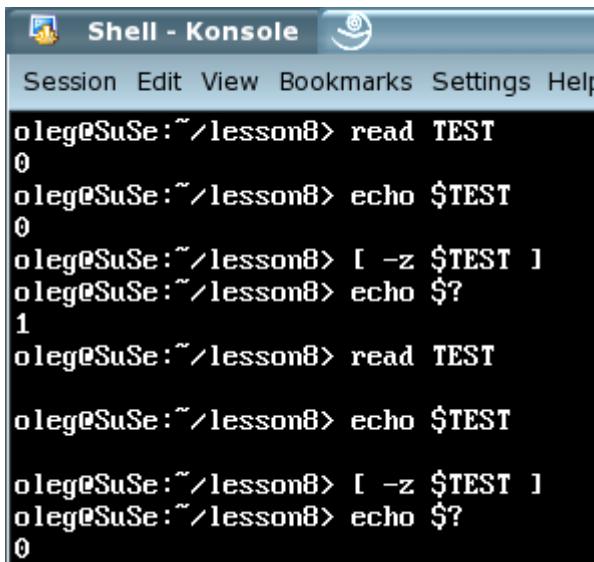
■ Киев    ■ Днепропетровск    ■ Львов    ■ Ровно    ■ Мариуполь    ■ Полтава  
■ Одесса    ■ Донецк    ■ Николаев    ■ Запорожье    ■ Луганск    ■ Харьков



---

**Оператор    Объяснение**

- =      Две строки равны
  - !=     Две строки не равны
  - z     Эта строка нулевая
  - n     Это строка не является нулевой
- 



```

Shell - Konsole
Session Edit View Bookmarks Settings Help
oleg@SuSe:~/lesson8> read TEST
0
oleg@SuSe:~/lesson8> echo $TEST
0
oleg@SuSe:~/lesson8> [ -z $TEST ]
oleg@SuSe:~/lesson8> echo $?
1
oleg@SuSe:~/lesson8> read TEST
oleg@SuSe:~/lesson8> echo $TEST
oleg@SuSe:~/lesson8> [ -z $TEST ]
oleg@SuSe:~/lesson8> echo $?
0

```

## 9.2.4 Проверка чисел

Операторами чисел могут быть:

---

**Оператор    Объяснение**

- eq     Два числа равны
  - ne     Два числа не равны
  - gt     Первое число больше второго
  - lt     Первое число меньше второго
  - le     Первое число меньше или равно второму
  - ge     Первое число больше или равно второму
- 



## 9.3 Переменные

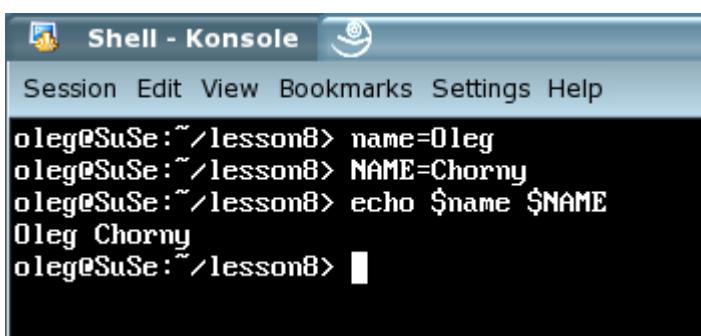
Чтобы продуктивно работать с интерпретатором, нужно уметь управлять его переменными. Переменными интерпретатора являются наименования, которым присваиваются значения. В качестве значений может выступать имя пути, имя файла или число. В любом случае интерпретатор воспринимает присвоенное значение как текстовую строку.

Существуют переменные двух типов - переменные интерпретатора (локальные переменные) и переменные среды. Переменные позволяют выполнить настройку среды. Они содержат информацию, которая применяется определенным пользователем. Благодаря этому система получает более подробные сведения о пользователях. Кроме того, переменные используются для сохранения констант.

### 9.3.1 Переменные интерпретатора

Переменные интерпретатора могут использоваться сценариями в период функционирования интерпретатора. После завершения выполнения интерпретатора действие этих переменных прекращается. Для присвоения определенного значения переменной воспользуйтесь конструкцией `имя_переменной=значение`.

Для отображения значений всех переменных интерпретатора воспользуйтесь командами `set` или `env`. Для отображения значений переменных воспользуйтесь конструкцией `echo $переменная1 $переменная2` и т.д.



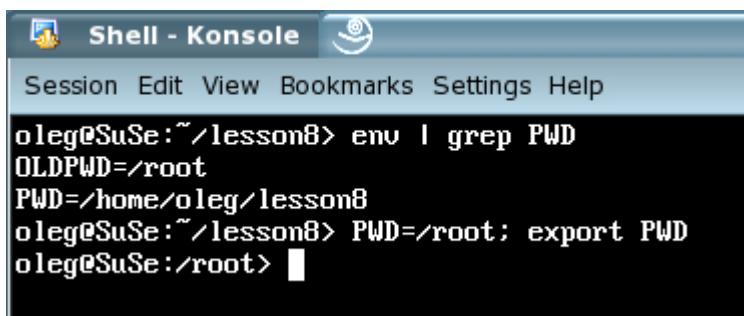
```
Shell - Konsole
Session Edit View Bookmarks Settings Help
oleg@SuSe:~/lesson8> name=Oleg
oleg@SuSe:~/lesson8> NAME=Chorny
oleg@SuSe:~/lesson8> echo $name $NAME
Oleg Chorny
oleg@SuSe:~/lesson8>
```



### 9.3.2 Переменные среды

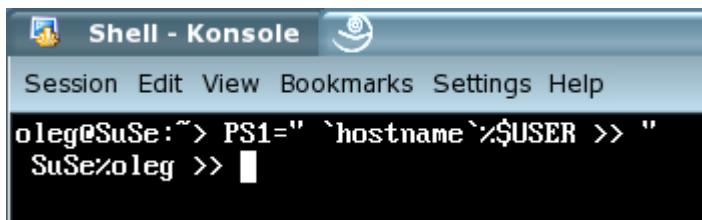
Переменные среды доступны для всех пользовательских процессов (часто их называют дочерними процессами). Пользователь регистрируется в процессе, который именуется родительским. Переменные среды доступны для всех дочерних процессов. Этими процессами могут быть редакторы, сценарии, приложения.

Переменные среды можно устанавливать в командной строке. Однако эти значения не сохраняются после выхода из системы. Поэтому переменные среды удобно инициализировать в пользовательском файле `~/.bashrc` или с помощью `rat_env.so`. Обычно при назначении переменных среды используют прописные буквы. Чтобы переменная была доступна всем процессам, достаточно применить команду `export`:



```
oleg@SuSe:~/lesson8> env | grep PWD
OLDPWD=/root
PWD=/home/oleg/lesson8
oleg@SuSe:~/lesson8> PWD=/root; export PWD
oleg@SuSe:/root>
```

Интерес представляет переменная `PS1`, задающая внешний вид приглашения командной строки:



```
oleg@SuSe:~> PS1="\`hostname\`\$USER > "
SuSe%oleg >
```

### 9.3.3 Позиционные и специальные переменные командной строки

Помимо переменных интерпретатора и переменных среды, имеются также позиционные и специальные переменные, которые предназначены только для чтения. Рассмотрим позиционные переменные. Они предназначены для сохранения



параметров, указанных в аргументах какой-либо команды, и имеют следующие названия:

\$1 \$2 \$3 ... \$9

Специальные переменные хранят в себе следующие параметры:

---

Переменная	Объяснение
------------	------------

---

\$#	Число аргументов передаваемых сценарию
\$*	В отдельной строке отображаются все аргументы, которые передаются сценарию. Здесь может содержаться более девяти параметров, в отличии от позиционных параметров
\$\$	Текущий идентификатор PID для выполняющегося сценария
\$!	Идентификатор PID для последнего процесса, который выполняется в фоновом режиме
\$@	Означает то же самое, что и параметр \$*
\$-	Отображение текущих опций интерпретатора команд, аналогично применению команды set
\$?	Показывает код завершения последней команды. Значение 0 свидетельствует об отсутствии ошибок, а любое другое значение - о их наличии.

---

Создадим сценарий по имени *test*, в котором будет следующее содержимое:

```
oleg@SuSe:~/lesson8> cat test
#!/bin/bash
echo "This is the script name: $0"
echo "This is the first parameter: $1"
echo "This is the second parameter: $2"
echo "This is the third parameter: $3"
echo "This is the fourth parameter: $4"
echo "This is the fifth parameter: $5"
echo "This is the sixth parameter: $6"
echo "This is the seventh parameter: $7"
echo "This is the eighth parameter: $8"
echo "This is the ninth parameter: $9"
echo "Show all parameters: $*"
echo "Any errors?: $?"
echo "Show my process ID: $$"
echo "Show number of arguments: $#"
```



Запустим созданный сценарий (предварительно сделав его исполняемым) и передадим сценарию ряд аргументов:

```
oleg@SuSe:~/lesson8> ./test Hello! How are You? My name is Oleg
This is the script name: ./test
This is the first parameter: Hello!
This is the second parameter: How
This is the third parameter: are
This is the fourth parameter: You?
This is the fifth parameter: My
This is the sixth parameter: name
This is the seventh parameter: is
This is the eighth parameter: Oleg
This is the ninth parameter:
Show all parameters: Hello! How are You? My name is Oleg
Any errors?: 0
Show my process ID: 3432
Show nuber of arguments: 8
oleg@SuSe:~/lesson8> █
```

Таким образом, данный пример демонстрирует то, как можно обращаться к содержимому позиционных и специальных переменных.

## 9.4 Управляющие конструкции

Все функциональные сценарии должны предлагать возможности по выбору возможных направлений своего развития.

### 9.4.1 Операторы if then else

Операторы *if then else* позволяют реализовать условное тестирование. Проверить условия можно самыми различными способами. Например, может производиться оценка размера файла, проверка установленных прав доступа к файлу, сравнение каких-либо числовых значений или строк.

В результате выполнения сравнений возвращается значение "истина" (0) или "ложь" (1), затем предпринимаются действия на основании полученного результата. Конструкция оператора if идеально подходит для проверки ошибок.

Этот оператор имеет следующий формат:

- Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава
- Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков



```

if условие
then
    команды 1
elseif условие 2
then
    команды 2
else
    команды 3
fi

```

Shell - Konsole

Session Edit View Bookmarks Settings Help

```

oleg@SuSe:~/lesson8> cat test
#!/bin/bash
echo -n "Enter your name: "
read NAME
if [ "$NAME" = "" ]
then
echo " You did not enter any information"
fi

oleg@SuSe:~/lesson8> ls -l test
-rwxr-xr-x 1 oleg users 121 2004-07-16 12:18 test
oleg@SuSe:~/lesson8> ./test
Enter your name:
 You did not enter any information
oleg@SuSe:~/lesson8> ./test
Enter your name: Oleg
oleg@SuSe:~/lesson8>

```

Сценарий, указанный на примере, осуществляет проверку того, ввел ли какое-либо имя пользователь. Если строка непустая – на экран выводится текст с именем. Если пустая – выводится текст, который сообщает о том, что пользователь ничего не вводил.

## 9.4.2 Оператор case

Оператор *case* является многовариантным оператором. С его помощью можно искать значения, используя заданный шаблон. Если совпадение с шаблоном установлено, можно выполнять команды, основываясь исключительно на этом соответствии.

Формат оператора *case* таков:

- Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава
- Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков



```

case значение in
шаблон1)
    команды1
    ...
;;
шаблон2)
    команды2
    ...
;;
esac

```

Shell - Konsole

Session Edit View Bookmarks Settings Help

```

oleg@SuSe:~/lesson8> cat test2
#!/bin/bash
echo -n "enter a number from 1 to 5: "
read ANS
case $ANS in
1) echo "you select 1"
;;
2) echo "you select 2"
;;
3) echo "you select 3"
;;
4) echo "you select 4"
;;
5) echo "you select 5"
;;
*) echo "`basename $0`: This is not between 1 and 5" >&2
exit 1
;;
esac

oleg@SuSe:~/lesson8> ./test2
enter a number from 1 to 5: 0
test2: This is not between 1 and 5
oleg@SuSe:~/lesson8> ./test2
enter a number from 1 to 5: 5
you select 5
oleg@SuSe:~/lesson8>

```

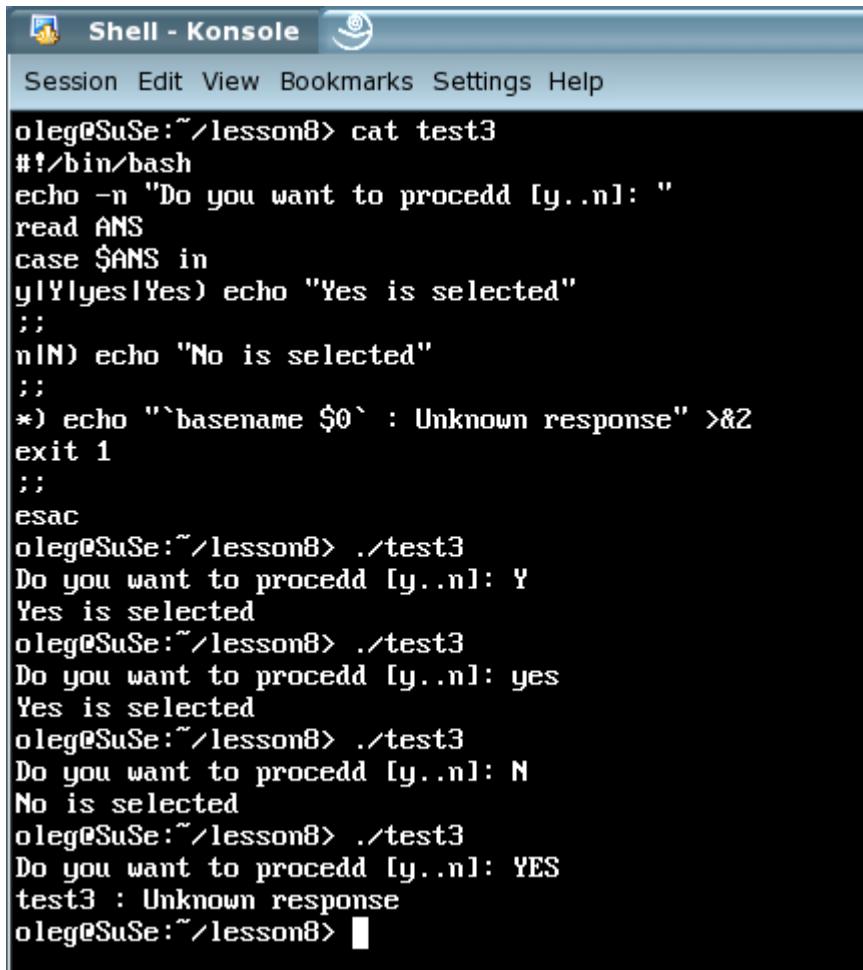
Сценарий проверяет, какое число ввел пользователь. Если пользователь вводит число не из запрашиваемого диапазона, сценарий сообщает пользователю об этом.

С помощью шаблона «\*)» выполняется прием информации, с которой не установлено соответствия.

- |          |                  |            |             |             |           |
|----------|------------------|------------|-------------|-------------|-----------|
| ■ Киев   | ■ Днепропетровск | ■ Львов    | ■ Ровно     | ■ Мариуполь | ■ Полтава |
| ■ Одесса | ■ Донецк         | ■ Николаев | ■ Запорожье | ■ Луганск   | ■ Харьков |



Оператор *case* удобно использовать для отображения запроса на продолжение обработки. При этом используется специальная конструкция при поиске по шаблону:



```

Shell - Konsole
Session Edit View Bookmarks Settings Help
oleg@SuSe:~/lesson8> cat test3
#!/bin/bash
echo -n "Do you want to procedd [y..n]: "
read ANS
case $ANS in
y|Y|yes|Yes) echo "Yes is selected"
;;
n|N|no|No) echo "No is selected"
;;
*) echo "`basename $0` : Unknown response" >&2
exit 1
;;
esac
oleg@SuSe:~/lesson8> ./test3
Do you want to procedd [y..n]: Y
Yes is selected
oleg@SuSe:~/lesson8> ./test3
Do you want to procedd [y..n]: yes
Yes is selected
oleg@SuSe:~/lesson8> ./test3
Do you want to procedd [y..n]: N
No is selected
oleg@SuSe:~/lesson8> ./test3
Do you want to procedd [y..n]: YES
test3 : Unknown response
oleg@SuSe:~/lesson8>

```

### 9.4.3 Цикл for

Общий формат цикла:

```

for имя_переменной in list
do
    команда1
    команда2
done

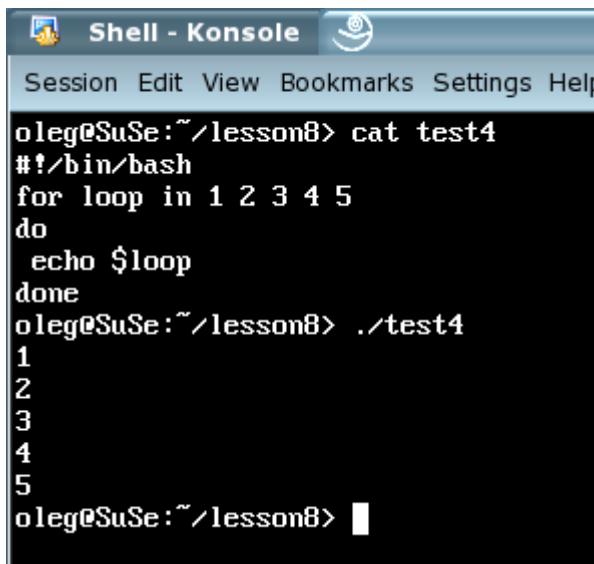
```

Цикл *for* однократно обрабатывает всю информацию для каждого значения, включенного в список *list*. Чтобы получить доступ к каждому значению в списке, достаточно задать параметр *имя\_переменной*. Командой служит любая действительная

- |          |                  |            |             |             |           |
|----------|------------------|------------|-------------|-------------|-----------|
| ■ Киев   | ■ Днепропетровск | ■ Львов    | ■ Ровно     | ■ Мариуполь | ■ Полтава |
| ■ Одесса | ■ Донецк         | ■ Николаев | ■ Запорожье | ■ Луганск   | ■ Харьков |

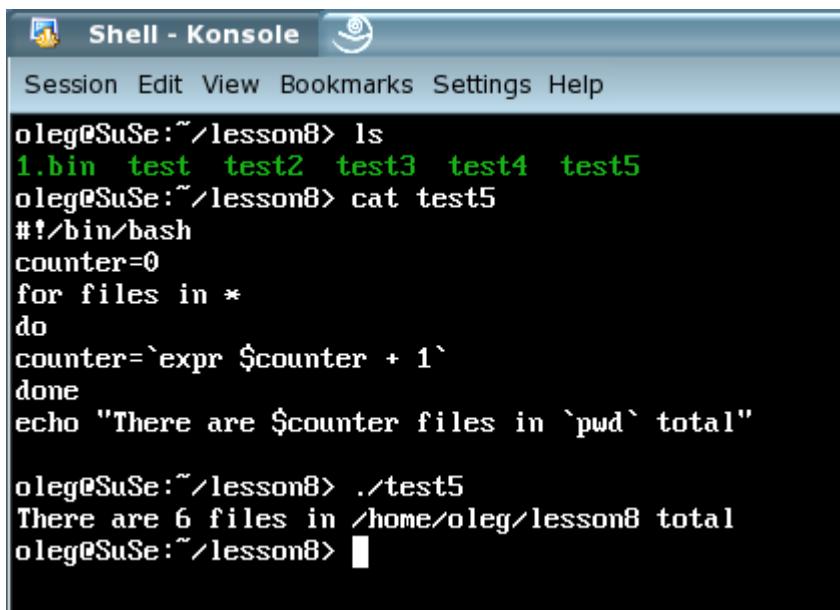


команда или оператор интерпретатора. В качестве параметра *имя\_переменной* можно указать любое слово. применение опции *in List* не является обязательным; если не включать эту часть, цикл воспользуется позиционными параметрами командной строки. Опция *in List* может содержать подстановки, строки и имена файлов.



```
oleg@SuSe:~/lesson8> cat test4
#!/bin/bash
for loop in 1 2 3 4 5
do
echo $loop
done
oleg@SuSe:~/lesson8> ./test4
1
2
3
4
5
oleg@SuSe:~/lesson8> █
```

Если необходимо ввести счетчик, пользуемся командой *expr*:



```
oleg@SuSe:~/lesson8> ls
1.bin  test  test2  test3  test4  test5
oleg@SuSe:~/lesson8> cat test5
#!/bin/bash
counter=0
for files in *
do
counter=`expr $counter + 1`
done
echo "There are $counter files in `pwd` total"
oleg@SuSe:~/lesson8> ./test5
There are 6 files in /home/oleg/lesson8 total
oleg@SuSe:~/lesson8> █
```





#### 9.4.4 Цикл until

Цикл *until* позволяет выполнять ряд команд, пока условие остается истинным. Практически цикл *until* противоположен по смыслу циклу *while*, который зачастую бывает более предпочтительным для использования. Формат цикла *until*:

*until* *условие*  
  *команда 1*

...  
*done*

Терминал

```

Файл Правка Вид Терминал Вкладки Справка
mail:~/Desktop # cat /tmp/scripts/script1.sh
#!/bin/bash
until [ "$ISROOT" ]
do
sleep 5
ISROOT=`who | grep student`
done
echo "student log in" | mail oleg
mail:~/Desktop # /tmp/scripts/script1.sh
mail:~/Desktop # su - oleg
Пароль:
oleg@mail:~> mail
Heirloom mailx version 12.2 01/07/07. Type ? for help.
"/var/spool/mail/oleg": 1 message 1 new
>N 1 root@mail.oleg.org Tue Nov 16 13:39 17/541
? 1
Message 1:
From root@mail.oleg.org Tue Nov 16 13:39:45 2010
X-Original-To: oleg
Delivered-To: oleg@mail.oleg.org
Date: Tue, 16 Nov 2010 13:39:44 +0200
To: oleg@mail.oleg.org
User-Agent: Heirloom mailx 12.2 01/07/07
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
From: root@mail.oleg.org (root)

student log in
?
```



Данный сценарий, при обнаружении пользователя *student* в системе сигнализирует об этом пользователю *oleg* с помощью электронного сообщения. Если пользователь *student* не обнаружен - цикл будет работать беспрерывно - и сообщение не будет отсылаться.

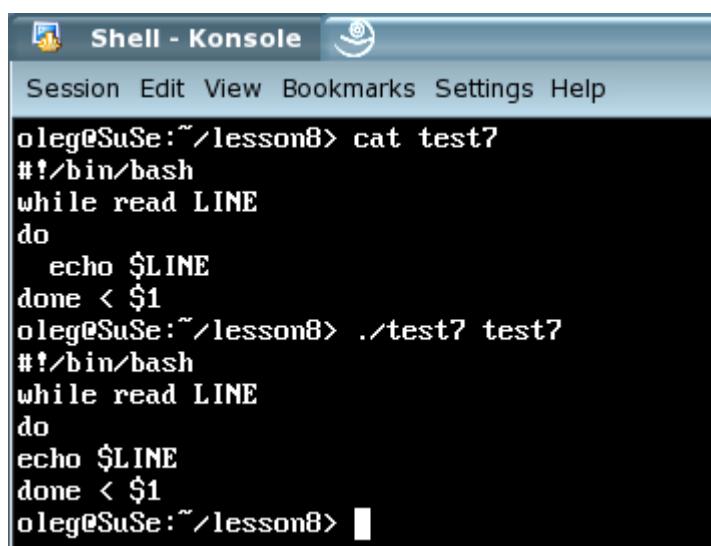
### 9.4.5 Цикл while

Цикл *while* выполняет ряд команд до тех пор, пока истинно условие. Формат цикла:

```
while условие
do
    команда 1
    ...
done
```

Команды, заключенные между *do* и *done* выполняются только в том случае, если код проверки условия равен нулю, иначе цикл заканчивается. Когда команды выполнены, контроль передается обратно, в верхнюю часть цикла. Все начинается снова, до тех пор, пока проверяемое условие не станет отличным от нуля.

С помощью данного цикла осуществим просмотр файла:



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
oleg@SuSe:~/lesson8> cat test7
#!/bin/bash
while read LINE
do
    echo $LINE
done < $1
oleg@SuSe:~/lesson8> ./test7 test7
#!/bin/bash
while read LINE
do
    echo $LINE
done < $1
oleg@SuSe:~/lesson8>
```

Данный сценарий принимает в параметрах командной строки имя файла, предназначенного для просмотра, и выводит его на экран. Для просмотра используются операции перенаправления потоков и переменная *LINE*, которая хранит в себе одну строку указанного файла.





Таким образом, использование командного интерпретатора является мощнейшим инструментом в руках системного администратора. Однако, не следует забывать о том, что этот инструмент может оказаться в руках злоумышленника, который может подменить своим сценарием какую-нибудь полюбившуюся вам команду. И в результате выполнения такой команды, вместо ожидаемого результата может произойти нечто непоправимое, особенно если вы использовали для запуска данной команды учетную запись *root*.

## 9.5 Подстановка имен файлов

При работе в режиме командной строки довольно много времени уходит на поиск необходимых файлов. Интерпретатор *bash* предлагает набор метасимволов, позволяющих находить файлы, имена которых соответствуют определенному шаблону.

---

Метасимвол	Назначение
------------	------------

---

- \* Соответствует произвольной строке, содержащей ноль и более символов
  - ? Соответствует любому символу
  - [...] Соответствует любому символу из числа заключенных в скобки
  - [!...] Соответствует любому символу за исключением тех, которые указаны в скобках
- 

Shell No. 3 - Konsole

```

Session Edit View Bookmarks Settings Help
oleg@SuSe:/etc> ls pass*
passwd passwd- passwd.old passwd.YaST2save
oleg@SuSe:/etc> ls *.old
group.old passwd.old shadow.old
oleg@SuSe:/etc> ls ??ss*
insserv.conf irssi.conf lesskey lesskey.bin passwd passwd- passwd.old passwd.YaST2save
oleg@SuSe:/etc> ls [ps]-
passwd- shadow-
oleg@SuSe:/etc> ls *[0-9].*
krb5.conf
oleg@SuSe:/etc> ls pass*.![o-y]*
passwd.YaST2save
oleg@SuSe:/etc> ls pass*.![o-Y]*
/bin/ls: pass*.![o-Y]*: No such file or directory
oleg@SuSe:/etc> ls [!i-l]?ss*.*
passwd.old passwd.YaST2save
oleg@SuSe:/etc>

```





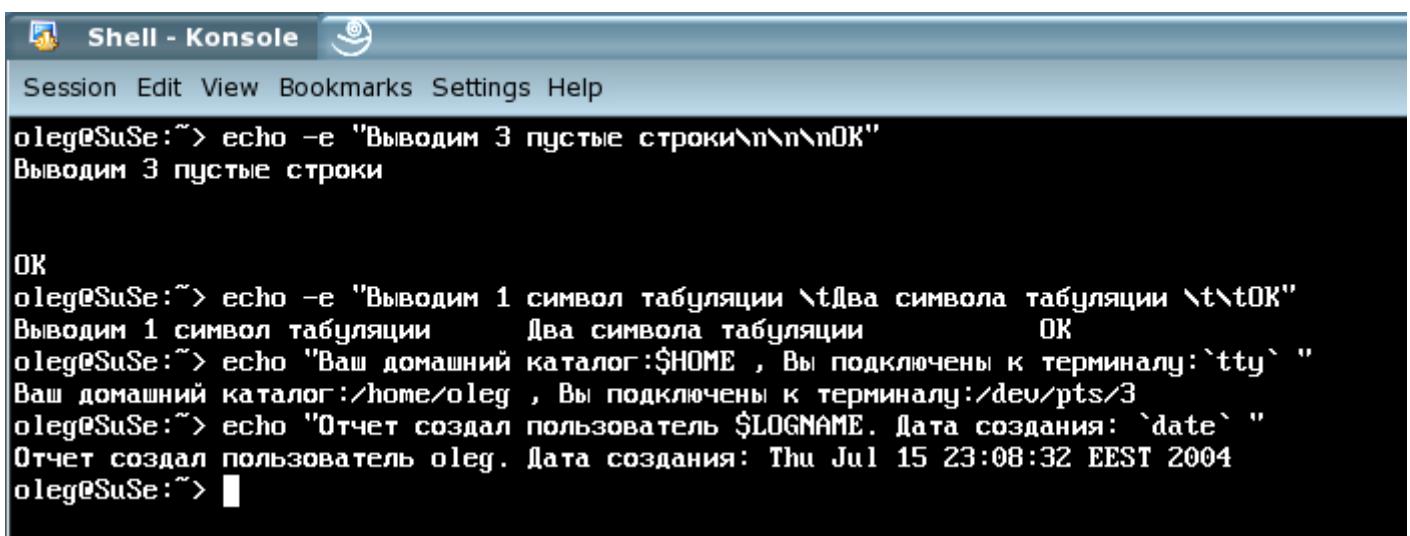
Обратите внимание на то, что символы «\n» и «\r» - интерпретируются как разные символы.

## 9.6 Ввод и вывод данных

Команды и сценарии могут получать входные данные двумя способами: из стандартного входного потока (связан с клавиатурой) или из файла. Аналогичное разделение существует и при выводе данных: результаты работы команды или сценария по умолчанию направляются на экран терминала, но можно перенаправить их в файл. Если в процессе работы возникают ошибки, сообщения о них тоже отображаются на экране. Чтобы этого избежать, нужно перенаправить поток ошибок в файл.

### 9.6.1 echo

Команда `echo` отображает на экране указанную строку текста. В строке могут встречаться управляющие символы, происходит обращение к переменным окружения, а также вызов команд:



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
oleg@SuSe:~> echo -e "Выводим 3 пустые строки\n\n\nOK"
Выводим 3 пустые строки

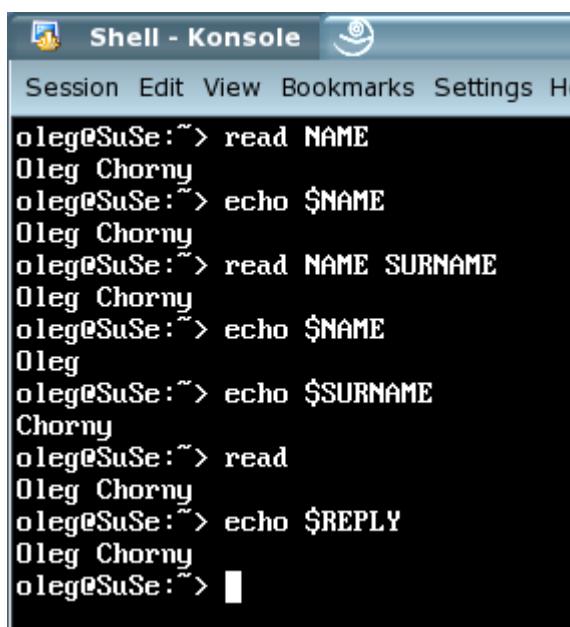
OK
oleg@SuSe:~> echo -e "Выводим 1 символ табуляции \tДва символа табуляции \t\tOK"
Выводим 1 символ табуляции      Два символа табуляции          OK
oleg@SuSe:~> echo "Ваш домашний каталог:$HOME , Вы подключены к терминалу:`tty` "
Ваш домашний каталог:/home/oleg , Вы подключены к терминалу:/dev/pts/3
oleg@SuSe:~> echo "Отчет создал пользователь $LOGNAME. Дата создания: `date` "
Отчет создал пользователь oleg. Дата создания: Thu Jul 15 23:08:32 EEST 2004
oleg@SuSe:~>
```

Также возможно применение управляющего символа `\c` - запрет отображения концевого символа новой строки. В отличие от Windows систем - конец строки и перевод на следующую строку - обозначаются одним символом.



## 9.6.2 read

Команда *read* читает одну строку из стандартного входного потока и записывает ее содержимое в указанные переменные. Если задана единственная переменная, в нее записывается вся строка. В результате ввода команды *read* без параметров строка перемещается в переменную среды *\$REPLY*. При указании нескольких переменных в первую из них записывается первое слово строки, во вторую - второе слово и т.д.



```
Session Edit View Bookmarks Settings Help
oleg@SuSe:~> read NAME
Oleg Chorny
oleg@SuSe:~> echo $NAME
Oleg Chorny
oleg@SuSe:~> read NAME SURNAME
Oleg Chorny
oleg@SuSe:~> echo $NAME
Oleg
oleg@SuSe:~> echo $SURNAME
Chorny
oleg@SuSe:~> read
Oleg Chorny
oleg@SuSe:~> echo $REPLY
Oleg Chorny
oleg@SuSe:~>
```

## 9.6.3 cat

Команда *cat* довольно проста, но универсальна. Эту команду удобно применять как для отображения файла, так и для его создания, а также при отображении файлов, содержащих управляющие символы.

Используя *cat*, следует помнить, что процесс вывода не приостанавливается по достижении конца страницы - файл пролистывается до конца.

С помощью данной команды также возможно "склеивать" части одного файла:

- Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава
- Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков





Shell - Konsole

Session Edit View Bookmarks Settings Help

```
oleg@SuSe:~/lesson8> echo Chorny > part1.txt
oleg@SuSe:~/lesson8> echo Oleg > part2.txt
oleg@SuSe:~/lesson8> echo Mihailovich > part3.txt
oleg@SuSe:~/lesson8> cat part1.txt part2.txt part3.txt >> FIO.txt
oleg@SuSe:~/lesson8> cat FIO.txt
Chorny
Oleg
Mihailovich
oleg@SuSe:~/lesson8> cat >> newfile.txt
Это новый файл. Таким образом, это еще один способ создания файла.
oleg@SuSe:~/lesson8> cat newfile.txt
Это новый файл. Таким образом, это еще один способ создания файла.
oleg@SuSe:~/lesson8>
```

## 9.6.4 Каналы

Каналом называется способ переадресации данных, при котором результаты работы одной команды передаются другой команде в виде входных данных. Канал организуется с помощью оператора «|». Например: `ls /etc | grep fstab` - в результате выполнения данной команды будет осуществлен вывод информации на экран только о файле `/etc/fstab`.

## 9.6.5 Стандартные потоки ввода, вывода, и ошибок

С каждым процессом (командой, сценарием и т.п.), выполняемым в интерпретаторе, связан ряд открытых файлов, из которых процесс может читать свои данные и в которые он может их записывать. Каждый из этих файлов идентифицируется числом, называемым дескриптором файла, но у первых трех файлов есть также имена, которые легче запомнить.

Стандартный поток ввода - дескриптор 0. По умолчанию входной поток ассоциирован с клавиатурой (устройство `/dev/tty`), но чаще всего он поступает по каналу от других процессов или из обычного файла. Стандартный поток вывода - дескриптор 1. В этот файл записываются все выходные данные процесса. По умолчанию данные выводятся на окно терминала (устройство `/dev/tty`), но их можно также перенаправить в файл или послать другому процессу. Стандартный поток

- |          |                  |            |             |             |           |
|----------|------------------|------------|-------------|-------------|-----------|
| ■ Киев   | ■ Днепропетровск | ■ Львов    | ■ Ровно     | ■ Мариуполь | ■ Полтава |
| ■ Одесса | ■ Донецк         | ■ Николаев | ■ Запорожье | ■ Луганск   | ■ Харьков |





ошибок - *дескриптор*. 2. В этот файл записываются сообщения об ошибках, возникающих в ходе выполнения каждой команды. По умолчанию сообщения об ошибках выводятся на экран терминала (*/dev/tty*), но их можно перенаправить в файл, организовывая, таким образом, эффективное ведение различного рода журнальной информации.

### 9.6.6 Файловый ввод-вывод

При вызове команд можно указывать, откуда следует принимать входные данные, куда необходимо направлять выходные данные, а также сообщения об ошибках. Интерпретатор позволяет использовать механизм переадресации, позволяющий ассоциировать стандартные потоки с различными файлами. Основные операторы переадресации:

Пример использования переадресации	Объяснение
команда > файл	Направляет стандартный поток вывода в указанный файл
команда 1> файл	Направляет стандартный поток вывода в указанный файл
команда >> файл	Направляет стандартный поток вывода в указанный файл (режим присоединения)
команда > файл 2>&1	Направляет стандартные потоки вывода и ошибок в указанный файл
команда 2> файл	Направляет стандартный поток ошибок в указанный файл
команда 2>> файл	Направляет стандартный поток ошибок в указанный файл (режим присоединения)
команда >> файл 2>&1	Направляет стандартные потоки вывода и ошибок в указанный файл (режим присоединения)
команда < файл1 > файл2	Получает входные данные из первого файла и направляет выходные данные во второй файл
команда < файл	В качестве стандартного входного потока использует данные из указанного файла

### 9.7 Порядок выполнения команд

При выполнении команд иногда требуется знать, как была выполнена другая команда. Предположим, что необходимо скопировать все файлы из одного каталога в другой, а



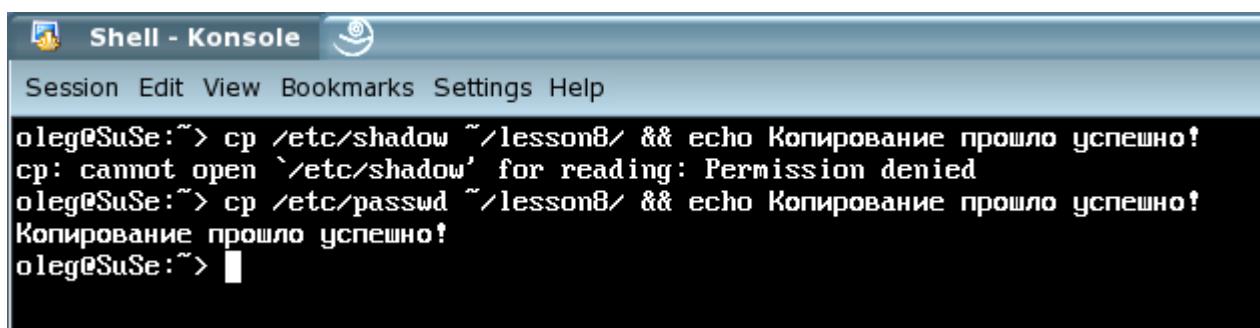
затем удалить исходный каталог. Прежде чем удалять исходный каталог, важно убедиться, что копирование прошло успешно. В противном случае содержимое исходного каталога может считаться утерянным.

### 9.7.1 Оператор &&

Общий формат оператора таков:

*команда1 && команда2*

Эта инструкция обрабатывается следующим образом: правый operand интерпретируется тогда, когда левый operand является истинным. Иными словами, вторая команда выполняется в том случае, если первая успешно завершилась.



```
oleg@SuSe:~> cp /etc/shadow ~/lesson8/ && echo Копирование прошло успешно!
cp: cannot open `/etc/shadow' for reading: Permission denied
oleg@SuSe:~> cp /etc/passwd ~/lesson8/ && echo Копирование прошло успешно!
Копирование прошло успешно!
oleg@SuSe:~> █
```

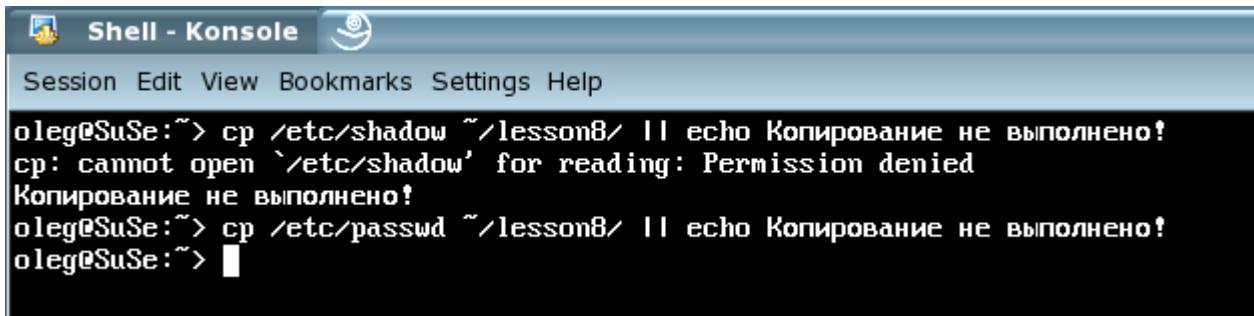
### 9.7.2 Оператор ||

Общий формат оператора таков:

*команда1 || команда2*

Эта инструкция обрабатывается следующим образом: правый operand интерпретируется тогда, когда левый operand является ложным. Иными словами, вторая команда выполняется в том случае, если первая завершилась неуспешно.





```
Session Edit View Bookmarks Settings Help
oleg@SuSe:~> cp /etc/shadow ~/lesson8/ || echo Копирование не выполнено!
cp: cannot open `/etc/shadow' for reading: Permission denied
Копирование не выполнено!
oleg@SuSe:~> cp /etc/passwd ~/lesson8/ || echo Копирование не выполнено!
oleg@SuSe:~>
```

### 9.7.3 Группирование команд

Существует возможность объединить несколько команд в группу и выполнить ее как единое целое в порожденном интерпретаторе. Для выполнения группы команд, следует перечислить их через точку с запятой и весь список заключить в скобки:

```
( cp /etc/passwd ~/Lesson8/ ; echo Копирование выполнено! )
```

## 9.8 Задания для самопроверки

- Пользователь *baduser* с завидным постоянством некорректно ведет себя в системе. При следующем его входе, предупредите данного пользователя о том, что в случае, если он будет продолжать себя плохо вести, его учетная запись будет заблокирована. Пусть он введет с клавиатуры подтверждение того, что он согласен с Вашиими условиями. Если он согласен - отослать пользователю *boss* текст Вашего соглашения с данным пользователем. Если не согласен - не впускать пользователя в систему.
- Вами был замечена деструктивная программа, признаками появления которой является наличие файла */tmp/viruson*. Необходимо осуществить защиту от данного вируса следующим образом:
  - при обнаружении данного файла, система в течение 10 секунд переводится в однопользовательский режим.
  - защита включается при переходе на 3 или 5 уровень исполнения автоматически.
- Вами было обнаружено, что использование программы *frozen-bubble* негативно сказывается на трудоспособности ваших коллег. Поэтому при каждом запуске данной программы - происходит уведомление Босса о том, что пользователи позволяют себе вольности во время работы, а также выключение данной программы через 30 секунд после ее запуска.



## 10. Выполнение заданий по расписанию

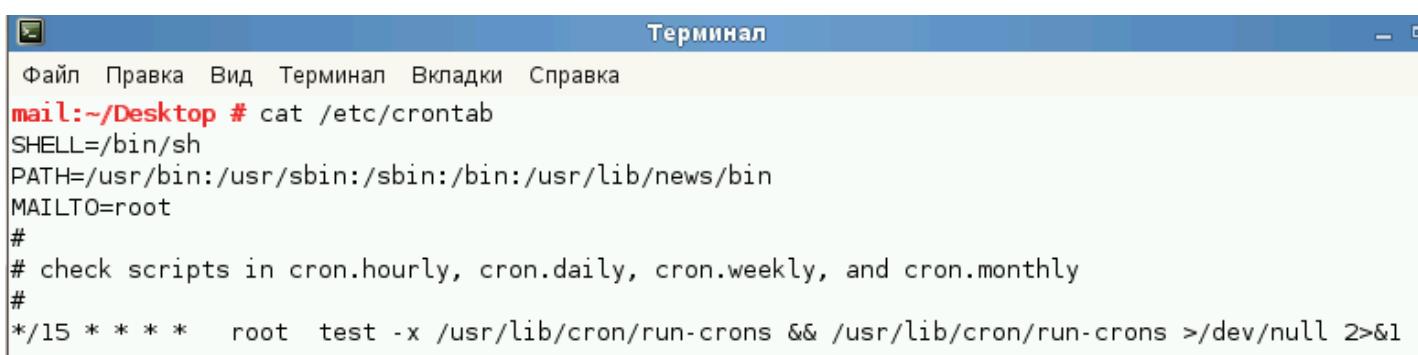
Одной из важнейших задач, которую требуется решить системному администратору, является максимальная автоматизация всех задач, которые необходимо выполнять в системе. Это позволит рационализировать работу системного администратора, высвободив для него больше свободного времени на рабочем месте. Благодаря этому, администратор будет способен больше времени уделять проектированию и внедрению новых решений, что благоприятно скажется на функционировании всего предприятия.

### 10.1 Планировщик cron

Программа *cron* является основным системным планировщиком, служащим для выполнения различных заданий в фоновом режиме. Команда *crontab* позволяет редактировать и удалять инструкции для программы *cron* посредством специального *crontab*-файла. У каждого пользователя может быть свой *crontab*-файл, но в крупных системах администратор обычно исключает такую возможность. В этом случае администратору необходимо создать вспомогательные файлы */etc/cron.deny* или */etc/cron.allow*, содержащие списки пользователей, которым соответственно запрещено и разрешено выполнять команду *crontab*. Для выполнения поставленных задач, должна быть запущена служба *cron*.

Обратите внимание на то, что необходимо создавать только один из данных файлов, в зависимости от того, какое правило по умолчанию используется на вашей системе. Если всем пользователям, за исключением некоторых, необходимо запретить использование *cron*, создается файл */etc/cron.allow*. Если наоборот - */etc/cron.deny*.

Файл */etc/crontab* используется как общесистемный файл расписания.



Терминал

```
mail:~/Desktop # cat /etc/crontab
SHELL=/bin/sh
PATH=/usr/bin:/usr/sbin:/sbin:/bin:/usr/lib/news/bin
MAILTO=root
#
# check scripts in cron.hourly, cron.daily, cron.weekly, and cron.monthly
#
*/15 * * * * root test -x /usr/lib/cron/run-crons && /usr/lib/cron/run-crons >/dev/null 2>&1
```





Формат файла:

минуты    часы    дни\_месяца    месяц    дни\_недели    пользователь    команда

Поле	Объяснение
Минуты	От 0 до 59
Часы	От 0 до 23
День месяца	От 1 до 31
Месяц	От 1 до 12
День недели	От 0 до 7 (0=7=воскресенье)
Пользователь	Пользователь, от имени которого будет выполнена команда
Команда	Команда, которая должна быть выполнена.

На примере указано задание, которое выполняется каждые 15 минут от имени пользователя *root*. В задании выполняется проверка наличия файла */usr/lib/cron/run-crons* и если он существует, выполняется его запуск. Различные пакеты, во время своей инсталляции, копируют свои скрипты в каталоги */etc/cron.hourly*, */etc/cron.daily*, */etc/cron.weekly* и */etc/cron.monthly*. Выполнение таких скриптов как раз и контролируется скриптом */usr/lib/cron/run-crons*.

Формат crontab-файла каждого пользователя идентичен формату файла */etc/crontab* за исключением того, что поле «пользователь» в нем не используется. Для добавления, редактирования или удаления записей в crontab-файле как правило используется редактор *vi*. Чтобы отредактировать файл, достаточно выполнить команду *crontab -e*. При сохранении файла, программа *cron* проверяет значения полей и информирует пользователя об ошибках. Если какая-либо запись содержит ошибку, файл не будет принят. Для удаления своего crontab-файла, используйте команду *crontab -r*, а для просмотра *crontab -l*.

Примеры записей в *crontab* файлах пользователей:

30 21 \* \* \* /sbin/shutdown -h 5 - Выключение компьютера каждый день в 21:30.  
 45 4 1,10,22 \* \* /usr/bin/updatedb - Выполнение обновлений базы утилиты locate в 4:45 утра 1-го, 10-го и 22-го числа каждого месяца.

- Киев
- Днепропетровск
- Львов
- Ровно
- Мариуполь
- Полтава
- Одесса
- Донецк
- Николаев
- Запорожье
- Луганск
- Харьков





`0,30 18-33 * * * /bin/echo `date` > /dev/console` - Вывод на экран текущей даты каждые 30 минут между 18:00 и 09:00 (позволяет узнать приблизительное время "зависания" системы).

Проинсталлированные crontab-файлы пользователей хранятся в каталоге `/var/spool/cron/tabs`. Имена данных файлов совпадают с именами пользователей, чье расписание заданий они хранят. Пользователь `root` может работать с crontab-файлами других пользователей, для этого в аргументах команды `crontab` необходимо использовать опцию «`-u username`». Например: `crontab -e -u student`.

## 10.2 at – однократное выполнение заданий

Команда `at` позволяет передавать задания демону `cron` для одноразового выполнения в назначенное время. Выдавая задание, команда `at` сохраняет в отдельном файле - как его текст, так и все текущие переменные среды. Заметим, что команда `crontab` не делает этого. По умолчанию все результаты выполнения задания направляются пользователю в виде электронного сообщения. Для успешной постановки задания в расписание, должна быть запущена служба `atd`.

Базовый формат команды таков:

`at [-f файл] [-l -d -m] время`

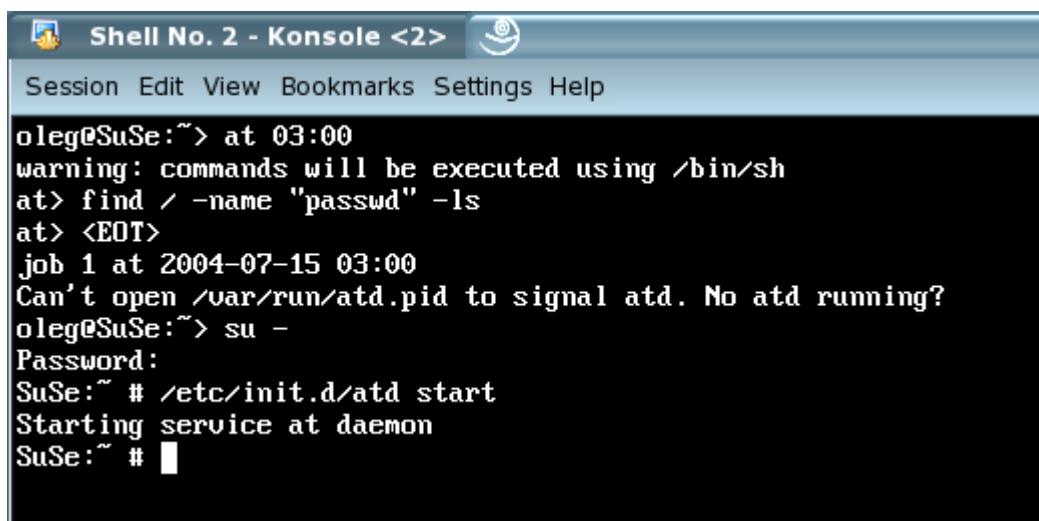
Аргумент	Назначение
<code>-f файл</code>	Список заданий должен быть взят из указанного файла
<code>-l</code>	Вывод на экран списка заданий, которые ожидают выполнения; аналогично команде <code>atq</code>
<code>-d</code>	Удаление задания с указанным номером; аналогично команде <code>atmr</code>
<code>-m</code>	Выдача пользователю сообщения электронной почты о завершении удаления
<b>Время</b>	Спецификация времени, когда будет выполнено задание. Эта спецификация может быть довольно сложной. Допускается указание не только времени в формате часы:минуты, но и даты, а также многочисленных ключевых слов, таких как названия дней недели, месяцы, наречий <code>today</code> (сегодня), <code>tomorrow</code> (завтра), <code>now</code> (сейчас). Наиболее удобна запись вида: <code>now + 3 hours</code> (через три часа)





Как и в случае с программой *cron*, пользователь *root* может контролировать, кому разрешено или запрещено выполнять команду *at*. Соответствующие списки пользователей содержатся в файлах */etc/at.allow* и */etc/at.deny*.

Текст задания можно передавать команде *at* двумя способами: в файле или в режиме командной строки *at*. В случае необходимости выполнения одиночной команды, вызовите *at* с указанием требуемого времени. В отображаемом приглашении необходимо ввести свою команду, а затем последовательно нажать клавиши *Enter* и *Ctrl+D*:



```
Session Edit View Bookmarks Settings Help
oleg@SuSe:~> at 03:00
warning: commands will be executed using /bin/sh
at> find / -name "passwd" -ls
at> <EOT>
job 1 at 2004-07-15 03:00
Can't open /var/run/atd.pid to signal atd. No atd running?
oleg@SuSe:~> su -
Password:
SuSe:~ # /etc/init.d/atd start
Starting service at daemon
SuSe:~ #
```

Запись *EOT* появляется после нажатия *Ctrl+D*. Обратите внимание, что команда *at* присваивает заданию уникальный идентификатор 1. Результаты выполнения команды *find* будут направлены вам по электронной почте.

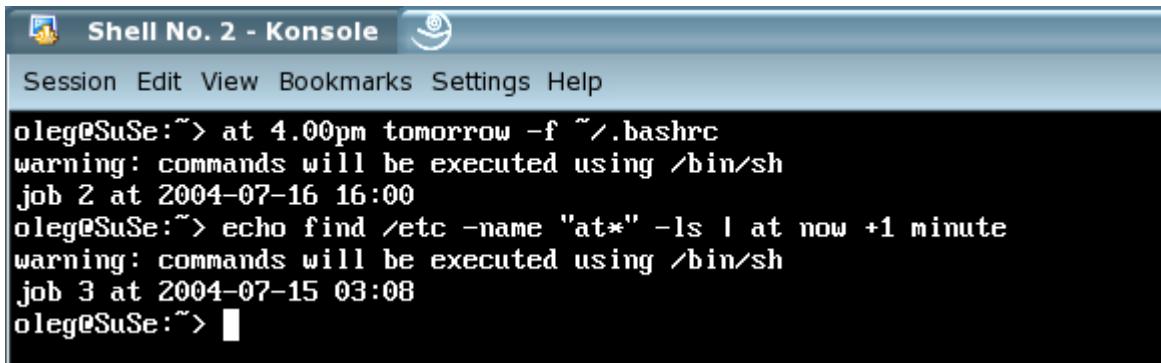
Обратите внимание на то, что после того как задание было добавлено в список, нас было проинформировано про то, что по всей видимости демон *atd* не запущен. Командой */etc/init.d/atd start* мы исправили это. Примеры указания корректного времени при вызове команды *at*:

- |                         |   |
|-------------------------|---|
| <i>at 6.45am May 12</i> | - 12 мая в 6:45 утра                                      |
| <i>at 11.10pm</i>       | - в 23:10 (сегодня или завтра, если это время уже прошло) |
| <i>at now + 1 hour</i>  | - через 1 час   |
| <i>at 9am tomorrow</i>  | - завтра в 9:00 утра                                      |
| <i>at 15:00 May 24</i>  | - 24 мая в 15:00  |
| <i>at 4am + 3 days</i>  | - через 3 дня в 4:00 утра                                 |



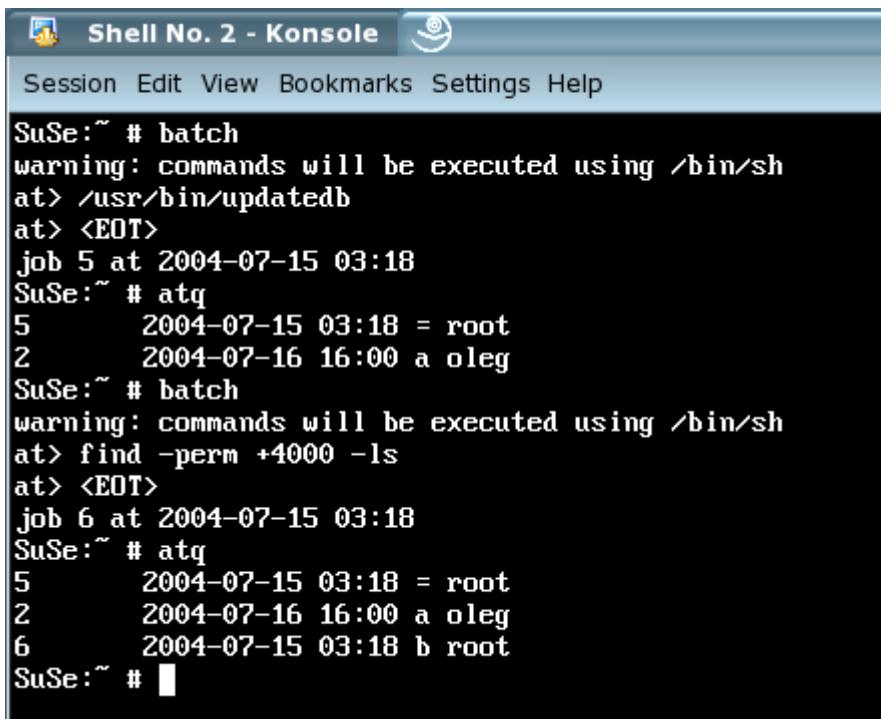


Если необходимо запустить с помощью *at* файл сценария, укажите его имя после опции *-f*. Так же задание можно передать с помощью команды *echo*:



```
oleg@SuSe:~> at 4.00pm tomorrow -f ~/.bashrc
warning: commands will be executed using /bin/sh
job 2 at 2004-07-16 16:00
oleg@SuSe:~> echo find /etc -name "at*" -ls | at now +1 minute
warning: commands will be executed using /bin/sh
job 3 at 2004-07-15 03:08
oleg@SuSe:~>
```

Существует также команда *batch*, которая планирует выполнение задания в период наименьшей загруженности системы. При просмотре всех заданий, такие команды помечаются символом «*b*», обычные команды – символом «*a*» (Просмотреть список выполненных заданий можно с помощью команды *atq*):



```
SuSe:~ # batch
warning: commands will be executed using /bin/sh
at> /usr/bin/updatedb
at> <EOT>
job 5 at 2004-07-15 03:18
SuSe:~ # atq
5      2004-07-15 03:18 = root
2      2004-07-16 16:00 a oleg
SuSe:~ # batch
warning: commands will be executed using /bin/sh
at> find -perm +4000 -ls
at> <EOT>
job 6 at 2004-07-15 03:18
SuSe:~ # atq
5      2004-07-15 03:18 = root
2      2004-07-16 16:00 a oleg
6      2004-07-15 03:18 b root
SuSe:~ #
```

Задание, выполняющееся в текущий момент, отмечается знаком «=».



## 10.3 Задания для самопроверки

1. Отключите пользователя *student* на три дня.
2. Разрешите пользователю *student* по четным часам входить в систему только с *tty2*, а по нечетным – только с *tty1*.
3. Ограничьте время входа всех непrivилегированных пользователей системы рабочими часами (Пн-Пт, с 9:00 до 18:00).
4. Разрешите использование программы Mozilla Firefox только по выходным дням.
5. В течение рабочего дня, в период с 9:00 до 18:00 пользователи не могут воспользоваться командой выключения питания ПК или просто остановки системы. После окончания рабочего времени, любой пользователь из зарегистрированных на тот момент в системе, может выключить ПК, но не ранее, чем через 5 минут после указания команды выключения питания. При этом если в системе находится Босс, то такая команда выключения питания автоматически отменяется, с указанием причины отмены пользователю, инициировавшему выключение.



## 11. Логирование событий

В основе журнальной регистрации событий во всех UNIX системах лежит система *Syslog*, которая принимает поступающие журнальные сообщения и обрабатывает их в соответствии с инструкциями, содержащимися в конфигурационном файле службы логирования.

Журнальные сообщения генерируются программами, службами и ядром. Это относится к большинству внутренних системных утилит, в частности к подсистемам электронной почты и печати, а также к внешним программам, таким как SSH. В терминологии системы *Syslog* программные компоненты, генерирующие сообщения, называются *средствами (facility)*. Каждое сообщение имеет определенный *уровень важности (level)*. В конфигурационном файле *Syslog* указано, как обрабатывать сообщения в зависимости от типа средства и уровня важности: записывать в файл, направлять в другую систему, посыпать пользователю или игнорировать.

### 11.1 Средства

Каждой службе, каждой программе и системной утилите соответствует определенное средство. Например:

Средство	Описание
<i>authpriv</i>	Сообщения, связанные с аутентификацией и содержащие конфиденциальную информацию
<i>cron</i>	Сообщения от службы cron и команды at
<i>daemon</i>	Сообщения различных служб (демонов)
<i>kern</i>	Сообщения ядра
<i>lpr</i>	Сообщения подсистемы печати
<i>mail</i>	Сообщения подсистемы электронной почты
<i>user</i>	Любые сообщения генерируемые пользовательскими программами
<i>Local0-Local7</i>	Используются программами, допускающими настройку журнальной регистрации
<i>news</i>	Сообщения сервера новостей



## 11.2 Уровни важности сообщений

Допустимые уровни, в порядке убывания степени важности:

Уровень	Описание
<i>emerg</i>	Работа системы парализована
<i>alert</i>	Ситуация, требующая немедленного реагирования
<i>crit</i>	Ошибка, препятствующая нормальной работе определенной утилиты или службы
<i>err</i>	Ошибка, препятствующая нормальной работе определенного компонента утилиты или службы
<i>warn</i>	Предупреждение
<i>notice</i>	Обычное сообщение, заслуживающее внимания
<i>info</i>	Информационное сообщение
<i>debug</i>	Вспомогательное сообщение, не связанное с ошибкой

Уровень важности *emerg* является самым высоким, уровень *alert* - второй по важности и т.д. Если в конфигурационном файле *Syslog* указан конкретный уровень, то подразумеваются сообщения этого и всех более высоких уровней.

## 11.3 Syslog-ng

Классической системой логирования сообщения в UNIX-системах является служба *Syslog*. Однако, во многих современных дистрибутивах ей на смену пришла улучшенная версия - *Syslog-ng*. Для работы службы ее нужно запустить: */etc/init.d/syslog start*.

### 11.3.1 /etc/syslog-ng/syslog-ng.conf

Данный файл является основным конфигурационным файлом службы *Syslog-ng*. Логирование производится директивами следующего вида:

```
Log { source(имя_источника); filter(имя_фильтра); destination(имя_приемника); };
```

Назначение данных директив:

- Киев   ■ Днепропетровск   ■ Львов   ■ Ровно   ■ Мариуполь   ■ Полтава
- Одесса   ■ Донецк   ■ Николаев   ■ Запорожье   ■ Луганск   ■ Харьков






---

<b>Директива</b>	<b>Назначение</b>
------------------	-------------------

---

<i>source</i>	Источник сообщений (локальный или удаленный)
<i>filter</i>	Фильтр сообщений по уровням важности сообщений, средствам сообщений, именам программ, или регулярным выражениям
<i>destination</i>	Куда будут направлены сообщения (файл, консоль, tcp/udp соединение, внешняя программа)
<i>Log</i>	Объединяет три выше указанных директивы в одну конструкцию

---

Источник сообщений может быть создан следующим образом:

*source имя\_источника { параметр1; параметр2; ...; };*

В качестве параметров директивы *source* может принимать:

Параметр	Описание
<i>file(имя файла)</i>	Считывать сообщения из указанного файла
<i>unix-dgram(имя файла)</i>	Считывать сообщения из указанного сокета (BSD стиль)
<i>unix-stream(имя файла)</i>	Считывать сообщения из указанного сокета (Linux стиль)
<i>udp(ip(адрес) port(порт))</i>	Адрес локального сетевого интерфейса, на котором производится сбор информации. UDP протокол. Для прослушивания всех интерфейсов – используется адрес 0.0.0.0
<i>tcp(ip(адрес) port(порт))</i>	Аналогично предыдущему, TCP протокол.
<i>internal()</i>	Внутренние сообщения Syslog-ng

---

Приемник сообщений создается следующим образом:

*destination имя\_приемника { параметр1; параметр2; ...; };*

В качестве параметров директивы *destination* может принимать:



Параметр	Описание
<i>file(имя файла)</i>	Записывать сообщения в указанный файл
<i>unix-dgram(имя файла)</i>	Записывать сообщения в указанный сокет (BSD стиль)
<i>unix-stream(имя файла)</i>	Записывать сообщения в указанный сокет (Linux стиль)
<i>udp(ip(адрес) port(порт))</i>	Адрес сетевого интерфейса, куда необходимо отправлять сообщения. UDP протокол.
<i>tcp(ip(адрес) port(порт))</i>	Аналогично предыдущему, TCP протокол.
<i>usertty(пользователь)</i>	Посыпает сообщения на пользовательский терминал

Фильтры создаются аналогичным образом:

```
filter имя_фильтра { выражение; };
```

Для создания фильтров можно использовать следующие выражения:

Параметр	Описание
<i>facility()</i>	Список средств, разделенных запятыми
<i>level()</i>	Список уровней важности сообщений
<i>program()</i>	Регулярное выражение для соответствия именам программ
<i>host()</i>	Регулярное выражение для соответствия именам хостов
<i>match()</i>	Регулярное выражение для соответствия именам программ

Рассмотрим примеры:

```
filter f_console { level(warn) and facility(kern) and not filter(f_ipTables)
                  or level(err) and not facility(authpriv); };
```

Фильтр *f\_console* «собирает» любые сообщения, начиная с уровня важности *err*. За исключением нескольких пунктов: для сообщений ядра собираются сообщения,





начиная с уровня *warn*, сообщения от брандмауэра *iptables* и сообщения об аутентификации не включаются в данный фильтр.

```
filter f_messages { not facility(news, mail) and not filter(f_iptables); };
```

Фильтр *f\_message* «собирает» любые сообщения, кроме сообщений брандмауэра, служб новостей и электронной почты.

```
source src {  
    #  
    # include internal syslog-ng messages  
    # note: the internal() source is required!  
    #  
    internal();  
  
    #  
    # the default log socket for local logging:  
    #  
    unix-dgram("/dev/log");  
  
    #  
    # uncomment to process log messages from network:  
    #  
    #udp(ip("0.0.0.0") port(514));  
};
```

В качестве источника *src* определены все сообщения, поступающие на сокет */dev/log* и сообщения самой системы *syslog-ng*.

```
#  
# All messages except iptables and the facilities news and mail:  
#  
destination messages { file("/var/log/messages"); };  
log { source(src); filter(f_messages); destination(messages); };
```

В первой строке создается приемник сообщений *f\_messages*, сообщения будут записываться в файл */var/log/messages*. Во второй строке указано, что все сообщения из источника *src*, отфильтрованные фильтром *f\_messages* следует отправлять приемнику сообщений *messages*.

## 11.4 Задания для самопроверки

1. Настройте логирование таким образом, чтобы все сообщения об аутентификации пользователей сохранялись в файле */var/log/auth*.



## 12. Файловая система /proc

Файловую систему */proc* можно рассматривать как интерфейс доступа к внутренним структурам данных ядра *Linux*. Ее можно использовать как для получения системной информации, так и для изменения некоторых параметров ядра во время работы системы и без ее перезагрузки. В зависимости от своего назначения, файл может быть доступен только для чтения или для чтения и редактирования.

### 12.1 Сбор системной информации

Рассмотрим часть файловой системы */proc*, доступную только для чтения.

#### 12.1.1 Каталог /proc/PID

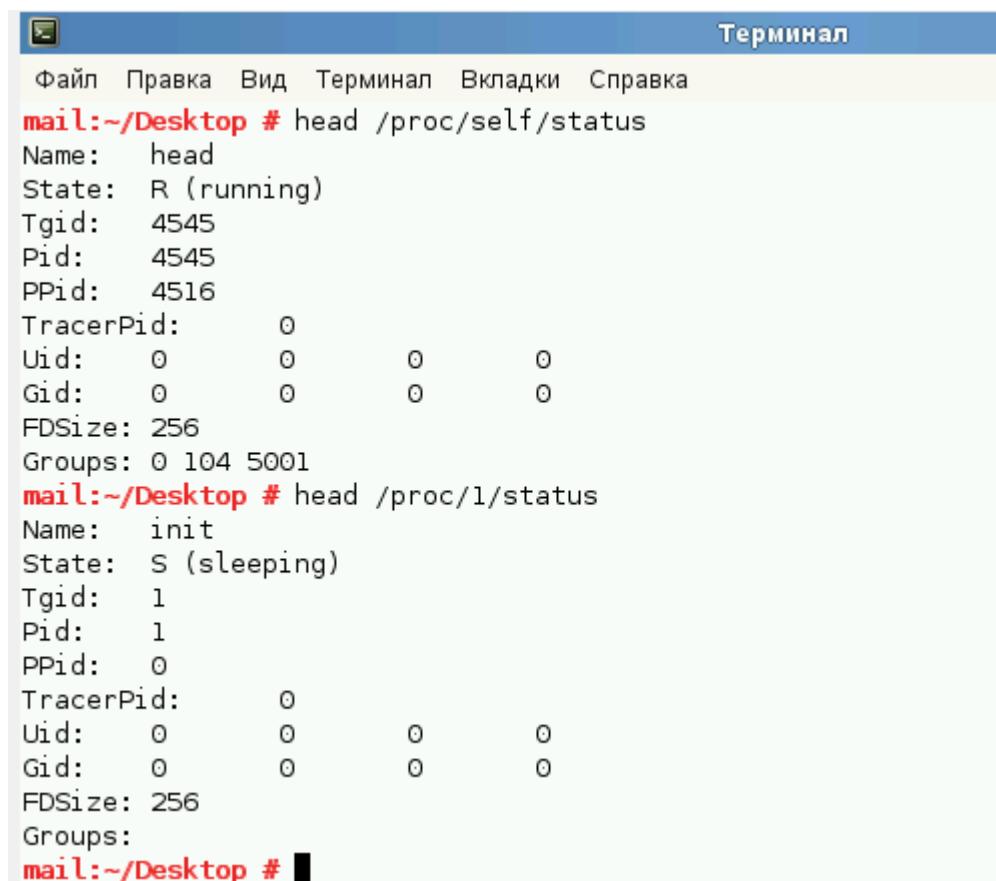
Каталог */proc* содержит, среди прочего, один подкаталог для каждого процесса, запущенного в системе. Имя каталога соответствует идентификатору процесса (*PID*).

Мягкая ссылка *self* ссылается на процесс, который обращается к файловой системе. Каждый подкаталог, соответствующий определенному процессу, хранит ряд файлов, в которых хранится определенная информация. Вот некоторые из них:

Файл	Назначение
<i>cmdline</i>	Аргументы командной строки
<i>cwd</i>	Ссылка на текущий рабочий каталог процесса
<i>environ</i>	Значения переменных окружения
<i>exe</i>	Ссылка на исполняемый файл процесса
<i>mem</i>	Память, используемая процессом
<i>root</i>	Ссылка на корневой каталог данного процесса
<i>stat</i>	Состояние процесса
<i>status</i>	Состояние процесса в читабельной форме



Для просмотра информации о процессе достаточно просмотреть соответствующий файл в подкаталоге процесса. Например, с помощью команды *head* просмотрим первые десять строк информации о состоянии процесса *init* и аналогичную информацию о собственном процессе, используя подкаталог *self*.



```
Файл Правка Вид Терминал Вкладки Справка
mail:~/Desktop # head /proc/self/status
Name: head
State: R (running)
Tgid: 4545
Pid: 4545
PPid: 4516
TracerPid: 0
Uid: 0 0 0 0
Gid: 0 0 0 0
FDSize: 256
Groups: 0 104 5001
mail:~/Desktop # head /proc/1/status
Name: init
State: S (sleeping)
Tgid: 1
Pid: 1
PPid: 0
TracerPid: 0
Uid: 0 0 0 0
Gid: 0 0 0 0
FDSize: 256
Groups:
mail:~/Desktop #
```

Информация, отображаемая в данных файлах, совпадает с той информацией, которую мы можем получить утилитами *ps* или *top*. Фактически, данные утилиты также используют файловую систему */proc* для получения данной информации. Если нам необходимо более детальная информация, чем та, которая предоставляется утилитами *ps* и *top*, можно воспользоваться просмотром файла */proc/PID/status*.

### 12.1.2 Данные ядра

Аналогично информации о процессах, информация о запущенном ядре находится в различных файлах и каталогах файловой системы */proc*. Приведем некоторые из них:





Файл	Назначение
<i>bus</i>	Каталог, содержащий информацию, специфичную для шин PCI, USB и т.д.
<i>cmdline</i>	Аргументы, с которыми запущено ядро (параметры, которые ему были переданы при загрузке)
<i>cputinfo</i>	Информация о процессоре
<i>devices</i>	Доступные устройства (блочные и символьные)
<i>dma</i>	Используемые DMA каналы
<i>filesystems</i>	Поддерживаемые файловые системы
<i>fd</i>	Файловые дескрипторы процесса, т.е. какие файлы он использует
<i>fs</i>	Параметры файловых систем, например файловой системы NFS
<i>interrupts</i>	Используемые прерывания
<i>iomem</i>	Карта памяти
<i>ioports</i>	Используемые порты ввода/вывода
<i>kcore</i>	Образ ядра
<i>kmsg</i>	Сообщения ядра
<i>Loadavg</i>	Средняя загрузка за последние 1, 5 и 15 минут, число процессов и номер следующего процесса.
<i>meminfo</i>	Информация о памяти
<i>modules</i>	Список загруженных модулей
<i>mounts</i>	Смонтированные файловые системы
<i>net</i>	Каталог, с информацией о сетевых устройствах и протоколах
<i>partitions</i>	Таблица разделов, известная системе
<i>scsi</i>	Информация про SCSI устройства
<i>stat</i>	Общая статистика
<i>swaps</i>	Информация об использовании swap
<i>uptime</i>	Количество секунд со времени загрузки системы и количество секунд ее простоя
<i>version</i>	Версия ядра





## 12.2 /proc/sys

Каталог */proc/sys* предназначен для изменения параметров работы ядра «на лету», без перезагрузки системы. Безусловно, делать это нужно очень осторожно, так как редактирование данных в этом каталоге может привести к выходу системы из строя. На практике, прежде чем применять параметры к производственному серверу, следует протестировать изменения их на альтернативном сервере, убедиться в том, что все происходит так, как и планировалось вами.

Для изменения файлов в каталоге */proc/sys* можно воспользоваться простой командой *echo*. Для выполнения подобной операции необходимо иметь привилегии суперпользователя:

```
mc - ~/Desktop
Файл Правка Вид Терминал Вкладки Справка
mail:~/Desktop # cat /proc/sys/net/ipv4/ip_forward
0
mail:~/Desktop # echo 1 > /proc/sys/net/ipv4/ip_forward
mail:~/Desktop # cat /proc/sys/net/ipv4/ip_forward
1
mail:~/Desktop #
```

На примере включена маршрутизация пакетов между сетевыми интерфейсами нашей системы, путем установки значения «1» в файл */proc/sys/net/ipv4/ip\_forward*.

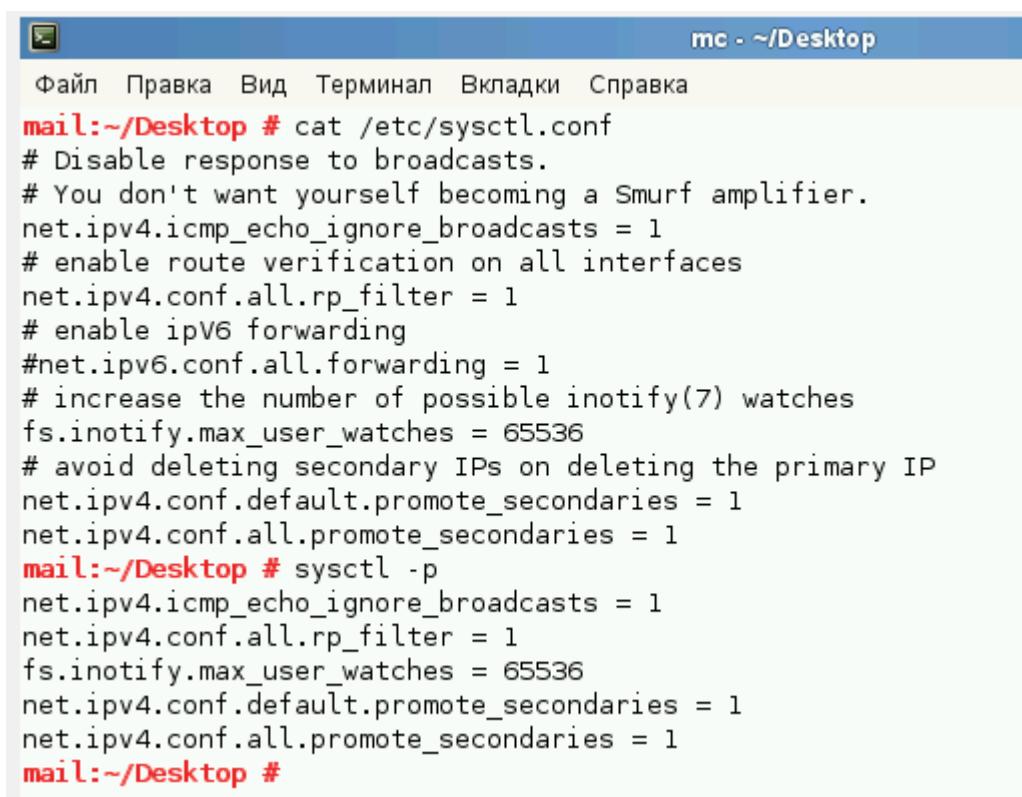
Альтернативный вариант – воспользоваться утилитой *sysctl*. Например:

```
mc - ~/Desktop
Файл Правка Вид Терминал Вкладки Справка
mail:~/Desktop # cat /proc/sys/net/ipv4/ip_forward
1
mail:~/Desktop # sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
mail:~/Desktop # sysctl -w net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
mail:~/Desktop # sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
mail:~/Desktop # cat /proc/sys/net/ipv4/ip_forward
0
mail:~/Desktop #
```



«Корневым» каталогом для утилиты *sysctl* является каталог */proc/sys*. Поэтому, имя необходимого параметра указывается относительно данного каталога. Имена каталогов и файлов разделяются друг от друга точками, а для присвоения определенного параметра используется оператор «=», после которого указывается необходимое значение.

У утилиты *sysctl* имеется свой конфигурационный файл - */etc/sysctl.conf*. Для применения указанных в нем параметров, необходимо воспользоваться следующей командой *sysctl -p*.



```
mc - ~/Desktop
Файл Правка Вид Терминал Вкладки Справка
mail:~/Desktop # cat /etc/sysctl.conf
# Disable response to broadcasts.
# You don't want yourself becoming a Smurf amplifier.
net.ipv4.icmp_echo_ignore_broadcasts = 1
# enable route verification on all interfaces
net.ipv4.conf.all.rp_filter = 1
# enable IPv6 forwarding
#net.ipv6.conf.all.forwarding = 1
# increase the number of possible inotify(7) watches
fs.inotify.max_user_watches = 65536
# avoid deleting secondary IPs on deleting the primary IP
net.ipv4.conf.default.promote_secondaries = 1
net.ipv4.conf.all.promote_secondaries = 1
mail:~/Desktop # sysctl -p
net.ipv4.icmp_echo_ignore_broadcasts = 1
net.ipv4.conf.all.rp_filter = 1
fs.inotify.max_user_watches = 65536
net.ipv4.conf.default.promote_secondaries = 1
net.ipv4.conf.all.promote_secondaries = 1
mail:~/Desktop #
```

Если необходимо, чтобы при старте системы автоматически изменялись определенные файлы в каталоге */proc/sys*, их можно указать в *sysctl.conf*. После этого следует в одном из стартовых скриптов вызвать команду *sysctl -p*.

## 12.3 Задания для самопроверки

1. Напишите свой сценарий, который принимает в качестве входного параметра PID любого запущенного процесса и сохраняет исполняемый файл данного процесса под именем */tmp/quarantine/PID-N*, где N – идентификатор процесса.



## 13. Настройка ядра

В 1960-е годы в Массачусетском Технологическом институте (MIT) и в ряде компаний была разработана экспериментальная операционная система Multics (Multiplexed Information and Computing Service) для машины GE-645. Один из разработчиков этой ОС, компания AT&T, отошла от Multics и в 1970 году разработала свою собственную систему Unics. Вместе с этой ОС поставлялся язык С. При этом С был разработан и написан так, чтобы обеспечить переносимость разработки операционной системы.

Двадцать лет спустя Эндрю Танненбаум (Andrew Tanenbaum) создал микроядерную версию UNIX® под названием MINIX (minimal UNIX), которая могла работать на небольших персональных компьютерах. Эта операционная система с открытым исходным кодом вдохновила Линуса Торвальдса (Linus Torvalds) на разработку первой версии Linux в начале 1990-х.

Linux быстро превратился из инициативы энтузиаста-одиночки во всемирный проект, в котором участвуют тысячи разработчиков. Одним из важнейших решений в судьбе Linux стало принятие лицензии GNU General Public License (GPL). GPL защитила ядро Linux от коммерческой эксплуатации и одновременно открыла путь к использованию разработок сообщества пользователей проекта GNU, основанного Ричардом Столлменом (Richard Stallman), объемы кода которого значительно превосходят даже объем ядра Linux. Это позволило использовать в Linux такие полезные приложения, как комплекс компиляторов GNU Compiler Collection (GCC) и различные командные оболочки.

С течением времени ядро Linux стало более эффективным с точки зрения использования памяти и процессорных ресурсов и приобрело исключительную стабильность. Однако самый интересный аспект Linux, учитывая размер и сложность этой системы - это ее переносимость. Linux можно откомпилировать для огромного количества разных процессоров и платформ, имеющих разные архитектурные ограничения и потребности.

Также следует отметить, что ядро Linux является динамическим (поддерживает добавление и удаление программных компонентов без остановки системы). Эти компоненты называются динамически загружаемыми модулями ядра. Их можно вводить в систему при необходимости, как во время загрузки (если найдено





конкретное устройство, для которого требуется такой модуль), так и в любое время по желанию пользователя.

Администратор системы имеет уникальную возможность самостоятельно настраивать ядро Linux. Принимать решение, какие элементы ядра должны быть входить в его монолитную часть, а какие – реализованы в виде модулей. Администратор может оптимизировать ядро под архитектуру своего процессора, избавиться от поддержки ненужных шин, файловых систем, сетевых протоколов. Все это может сказаться на быстродействие ядра, и, как следствие, приросте производительности системы.

### 13.1 Версии ядра

Линус Торвальдс продолжает выпускать новые версии ядра, объединяя изменения, вносимые другими программистами, и внося свои. Оно обычно называется «ванильным» (vanilla), то есть официальное ядро без каких-либо сторонних изменений. В дополнение к официальным версиям ядра существуют альтернативные ветки, которые могут быть взяты из различных источников. Как правило, разработчики дистрибутивов Linux поддерживают свои собственные версии ядра, например, включая в них драйверы устройств, которые ещё не включены в официальную версию.

Номер версии ядра Linux в настоящее время содержит четыре числа, следуя недавнему изменению в долго используемой до этого политике схемы версий, основанной на трёх числах. Для иллюстрации допустим, что номер версии составлен таким образом: A.B.C[.D] (например 2.2.1, 2.4.13 или 2.6.12.3).

Число А обозначает версию ядра. Оно изменяется менее часто и только тогда, когда вносятся значительные изменения в код и концепцию ядра. Оно изменилось дважды в истории ядра: в 1994 (версия 1.0) и в 1996 (версия 2.0).

Число В обозначает старшую версию ревизии ядра. Чётные числа обозначают стабильные ревизии, то есть те, которые предназначены для промышленного использования, такие как 1.2, 2.4 или 2.6. Нечётные числа обозначают ревизии для разработчиков, такие как 1.1 или 2.5. Они предназначены для тестирования новых улучшений и драйверов до тех пор, пока они не станут достаточно стабильными для того, чтобы быть включёнными в стабильный выпуск.





Число С обозначает младшую версию ревизии ядра. В старой трёхчисловой схеме нумерации, оно изменялось тогда, когда в ядро включались заплатки связанные с безопасностью, исправления ошибок, новые улучшения или драйверы. С новой политикой нумерации, однако, оно изменяется только тогда, когда вносятся новые драйверы или улучшения; небольшие исправления поддерживаются числом D.

Число D впервые появилось после случая, когда в коде ядра версии 2.6.8 была обнаружена грубая, требующая незамедлительного исправления ошибка, связанная с NFS. Однако, было недостаточно других изменений, для того чтобы это послужило причиной для выпуска новой младшей ревизии (которой должна была стать 2.6.9). Поэтому была выпущена версия 2.6.8.1 с единственным исправлением в виде исправления для этой ошибки. С ядра 2.6.11, эта нумерация была адаптирована в качестве новой официальной политики версий. Исправления ошибок и заплатки безопасности теперь обозначаются с помощью четвёртого числа, тогда как большие изменения выполняются в изменениях младшей версии ядра (число С).

## 13.2 Получение исходных кодов ядра

Для получения последней, самой актуальной версии ядра Linux, следует обращаться на сайт [www.kernel.org](http://www.kernel.org). Загрузите интересующую вас версию ядра и распакуйте его исходные коды в каталог `/usr/src`:

```
mail:/usr/src # wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.36.tar.bz2
--2010-11-22 00:54:20-- http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.36.tar.bz2
Распознаётся www.kernel.org... 204.152.191.37, 149.20.20.133
Устанавливается соединение с www.kernel.org|204.152.191.37|:80... соединение установлено.
Запрос HTTP послан, ожидается ответ... 200 OK
Длина: 70277083 (67M) [application/x-bzip2]
Сохраняется в каталог: `linux-2.6.36.tar.bz2'.

100%[=====] 70 277 083 241K/s   в 4m 45s

2010-11-22 00:59:05 (241 KB/s) - `linux-2.6.36.tar.bz2' сохранён [70277083/70277083]

mail:/usr/src # tar xjf linux
linux/          linux-2.6.27.19-5-obj/ linux-2.6.36.tar.bz2
linux-2.6.27.19-5/  linux-2.6.27.7-9-obj/ linux-obj/
mail:/usr/src # tar xjf linux-2.6.36.tar.bz2
mail:/usr/src # ls
linux           linux-2.6.27.7-9-obj  linux-obj        vboxvfs-3.0.8
linux-2.6.27.19-5  linux-2.6.36       packages        vboxvideo-3.0.8
linux-2.6.27.19-5-obj  linux-2.6.36.tar.bz2  vboxadd-3.0.8
```





Еще один способ получения исходных кодов – установка пакета *kernel-source*, входящего в дистрибутив.

```
Терминал
Файл Правка Вид Терминал Вкладки Справка
mail:/usr/src # rpm -qi kernel-source
Name        : kernel-source
Version    : 2.6.27.19
any
Release    : 5.1
Install Date: Срд 21 Окт 2009 20:08:30
Group      : Development/Sources
Size       : 310266257
Signature   : RSA/8, Сбт 28 Фев 2009 09:29:21, Key ID e3a5c360307e3d54
Packager   : http://bugs.opensuse.org
URL        : http://www.kernel.org/
Summary    : The Linux Kernel Sources
Description :
Linux kernel sources with many fixes and improvements.

Authors:
-----
Linus Torvalds <torvalds@osdl.org>
see /usr/src/linux/CREDITS for more details.

Source Timestamp: 2009-02-28 04:40:21 +0100
GIT Revision: 91f0acc0c21359752a2b781f5b1a002459e4c2a2
GIT Branch: SLE11_BRANCH
Distribution: SUSE Linux Enterprise 11
mail:/usr/src #
```

Какое из ядер настраивать в вашей системе? Принимайте решение самостоятельно. Но помните, что разработчики дистрибутива пристально следят за изменениями и новшествами в ядрах Linux. Тщательно тестируют новые ядра, применяют к ним свои изменения и после этого выкладывают в репозиториях в качестве патчей ядра, которые можно установить простым обновлением системы.

Однако если в вашей системе есть срочная необходимость в самых свежих драйверах определенных устройств, или в исправлениях ядра, которые критичны для работы вашей системы, вы можете, не дожидаясь реакции производителя дистрибутива, самостоятельно получить и установить самую свежую версию ядра.





## 13.3 Конфигурирование ядра

Для конфигурирования ядра, хорошим подспорьем может стать конфигурация текущего ядра, ядра, которое используется в нашей системе. Эта конфигурация хранится под именем `/boot/config-версия_ядра`. Скопируем текущую конфигурацию в каталог с исходными кодами ядра, только что полученного нами. Назначим этому конфигурационному файлу имя `.config`:

```
Терминал
Файл Правка Вид Терминал Вкладки Справка
mail:/usr/src/linux-2.6.36 # cp /boot/config-`uname -r` ./config
mail:/usr/src/linux-2.6.36 # ls
arch CREDITS firmware init lib mm samples tools
block crypto fs ipc .mailmap net scripts usr
.config Documentation .gitignore Kbuild MAINTAINERS README security virt
COPYING drivers include kernel Makefile REPORTING-BUGS sound
mail:/usr/src/linux-2.6.36 #
```

Теперь можно приступать к конфигурированию ядра, что можно проделывать разными способами (полный список можно получить командой `make help`):

Команда	Объяснение
<code>make config</code>	конфигурирование в текстовом режиме, про каждый элемент ядра будет задан вопрос: включить элемент в ядро, не включать, включить модульно.
<code>make menuconfig</code>	конфигурирование с использованием псевдографического меню
<code>make xconfig</code>	конфигурирование с использованием системы X-Windows и библиотеками Qt
<code>make gconfig</code>	конфигурирование с использованием системы X-Windows и библиотеками Gtk
<code>make oldconfig</code>	конфигуратор принимает ответы по умолчанию из ранее созданного <code>.config</code> -файла, и задает вопросы только касательно новых элементов
<code>make silentoldconfig</code>	действует по принципу предыдущего, только позволяет избежать загромождения экрана вопросами, на которые уже дан ответ
<code>make allyesconfig</code>	конфигуратор создает <code>.config</code> -файл, отмечая все значения как – Y, таким образом, все возможные элементы ядра будут включены в него монолитно
<code>make allmodconfig</code>	конфигуратор создает <code>.config</code> -файл, отмечая все значения как – M, таким образом все возможные элементы ядра будут скомпилированы как модули





Как правило, рекомендуется использовать команду *make menuconfig*. Для ее работы должен быть установлен пакет под именем *ncurses-devel*. Данная команда запустит псевдографический режим конфигурирования ядра. Для загрузки конфигурации можно использовать опцию *Load An Alternate Configuration File*.

Внесем изменения в текущую конфигурацию ядра. Например, отключим поддержку шины ISA:

```

Bus options (PCI etc.)
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are
hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press
<Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded
<M> module < > module capable
^(-)
[*] PCI Express support
<M> PCI Express Hotplug driver
[*] Root Port Advanced Error Reporting support
[ ] PCI Express ECRC settings control (NEW)
< > PCIe AER error injector support (NEW)
- *- PCI Express ASPM control
[ ] Debug PCI Express ASPM
[*] Message Signaled Interrupts (MSI and MSI-X)
[ ] PCI Debugging
< > PCI Stub driver (NEW)
[*] Interrupts on hypertransport devices
[ ] PCI IOV support (NEW)
[ ] ISA support
[ ] MCA support
<M> NatSemi SCx200 support
<M> NatSemi SCx200 27MHz High-Resolution Timer Support
[*] One Laptop Per Child support
v(+)

<Select> < Exit > < Help >

```

Наличие знака «\*» возле параметра – означает, что он включен в ядро. Наличие символа «M» - означает, что данный элемент ядра будет реализован в виде модуля. Отсутствие символов означает, что данный элемент будет отключен.

Для идентификации версий ядер и для успешной инсталляции ядра в случае, если будет создаваться *rpm* пакет, в разделе *General Setup --> (default) Local version* можно назначить новую версию конфигурируемого ядра новое имя:





Local version - append to kernel release

Please enter a string value. Use the <TAB> key to move from the input field to the buttons below it.

-5-test

< Ok > < Help >

Когда с закончено, необходимо выбрать *Exit*, и ответить на вопрос (*Do you wish to save your new kernel configuration?*) *Yes*. Конфигурация ядра сохранена в файле */usr/src/linux-версия\_ядра/.config*.

## 13.4 Сборка ядра

Сборку ядра можно осуществлять также несколькими способами. Например, командой *make rpm* можно создать RPM пакет с новой версией ядра. Более традиционный способ – такое последовательное выполнение команд:

```
make bzImage
make modules
make modules_install
make install
```

Первая команда создает сжатую версию сконфигурированного ядра, вторая – компилирует объекты, выбранные как модули, третья команда – инсталлирует модули в заданный каталог, четвертая – инсталлирует ядро с использованием сценария */sbin/installkernel*. Команда *make all* заменяет собой первые две команды.

Сценарий */sbin/installkernel* выполняет следующие действия: записывает в каталог */boot* образ ядра и файл *System.map*, там же создает *initrd* файл, конфигурируется соответствующим образом файл *menu.lst* загрузчика GRUB.

Перезагрузившись и выбрав в меню загрузки необходимый пункт, можно протестировать работу нового, самостоятельно собранного ядра. В случае непредвиденных неполадок с ним, есть возможность вернуться к прежнему ядру. Конечно, в таком случае нужно, чтобы устаревшие, но рабочие версии ядра были доступны в вашей системе. Не удаляйте их прежде времени.



## 14. Настройка сетевого подключения

Когда ядро обнаруживает сетевую карту, создается соответствующий ей сетевой интерфейс, которому назначается имя, зависящее от порядка обнаружения устройств и порядка загрузки модулей ядра. По умолчанию, только в простых системах можно с уверенностью предположить какое имя будет присвоено интерфейсу при следующей загрузке, так как в системах, которые допускают добавление или удаление устройств, не могут быть стабильны в именовании устройств.

Однако, все системные средства основаны на постоянных именах сетевых интерфейсов. Проблема решается с помощью *udev*. Генератор постоянных имен *udev* создает правила, с помощью которых каждому интерфейсу назначается постоянное имя, привязанное к MAC-адресу сетевой карты. База данных сетевых интерфейсов *udev* находится в файле */etc/udev/rules.d/70-persistent-net.rules*. Каждая строка данного файла символизирует собой один сетевой интерфейс. Назначенное имя можно изменить, исправив опцию *NAME=""* соответствующей строки.

### 14.1 Ручная настройка интерфейсов

Ручная настройка сетевого подключения в современных дистрибутивах не является необходимостью, большинство операций можно выполнить, используя графическую оболочку. Однако знание соответствующих конфигурационных файлов позволит с большей уверенностью управлять системой и в случае непредвиденных неисправностей – легче справляться с ними. Для управления интерфейсами используются следующие команды:

Команда	Описание
<i>ifup имя_интерфейса</i>	Осуществляет запуск интерфейса
<i>ifdown имя_интерфейса</i>	Осуществляет остановку интерфейса
<i>ifstatus имя_интерфейса</i>	Выводит на экран информацию об интерфейсе
<i>rcnetwork</i>	Используется для запуска, перезапуска и просмотра информации обо всех сетевых интерфейсах. Используются аргументы <i>start, stop, restart, status</i> . В аргументах можно указывать имя одного интерфейса. Например <i>rcnetwork restart eth0</i> .



## 14.1.1 /etc/sysconfig/network/ifcfg-\*

Эти файлы содержат конфигурации сетевых интерфейсов. Возможные опции:

Опция	Возможное значение
STARTMODE	manual – интерфейс будет запущен только если вручную запускается утилита <i>ifup</i> auto – интерфейс будет запущен при загрузке или при горячем подключении к системе off – никогда не будет активен
BOOTPROTO	static – используется фиксированный адрес dhcp – получать IP адрес от DHCP сервера
IPADDR	IPv4 или IPv6 адрес. Для указания нескольких адресов необходимо указать данную переменную несколько раз, используя в каждом случае разный суффикс
PREFIXLEN	Количество бит в IPADDR, использующиеся для адреса сети.
NETMASK	Сетевая маска. Менее приоритетная опция, чем опция PREFIXLEN
BROADCAST	Широковещательный адрес сети. Если не указывать, будет вычислен автоматически
MTU	MTU данного интерфейса
ETHTOOL_OPTIONS	Если данная опция заполнена, <i>ifup</i> вызывает <i>ethtool</i> с указанными опциями.

Терминал

```

Файл Правка Вид Терминал Вкладки Справка
mail:~/Desktop # cat /etc/sysconfig/network/ifcfg-eth0
BOOTPROTO='static'
BROADCAST=''
ETHTOOL_OPTIONS=''
IPADDR='192.168.40.1'
MTU=''
NAME='79c970 [PCnet32 LANCE]'
NETMASK='255.255.255.0'
NETWORK=''
REMOTE_IPADDR=''
STARTMODE='auto'
USERCONTROL='no'
mail:~/Desktop # rcnetwork status eth0
    eth0      device: Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE] (rev 40)
        eth0      IP address: 192.168.40.1/24
Checking service network . . . .
mail:~/Desktop #

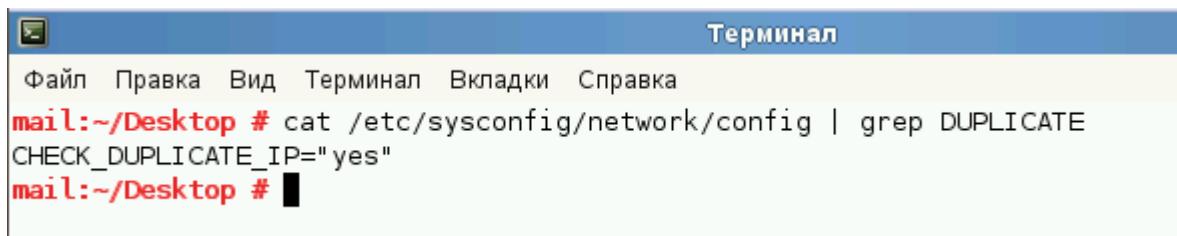
```

running  
running



## 14.1.2 /etc/sysconfig/network/config

Данный файл задает общие настройки работы утилит *ifup*, *ifdown*, *ifstatus*. Содержимое данного файла содержит подробные комментарии о назначении опций, используемых в данном файле. Например, опция *CHECK\_DUPLICATE\_IP* позволяет указывать, следует ли перед назначением IP адреса интерфейсу выполнять ARP-запросы в сеть для получения информации о том, не занят ли данный IP адрес каким-либо узлом в сети. Такая проверка добавит порядка одной секунды ко времени применения настроек к каждому интерфейсу. В случае если в системе используется немало сетевых интерфейсов, такая проверка может быть лишней. По этой причине такая проверка, по умолчанию, не производится. Для многих администраторов это становится настоящим сюрпризом. Для изменения такого поведения, присвойте переменной *CHECK\_DUPLICATE\_IP* значение “yes”:



```
Файл Правка Вид Терминал Вкладки Справка
mail:~/Desktop # cat /etc/sysconfig/network/config | grep DUPLICATE
CHECK_DUPLICATE_IP="yes"
mail:~/Desktop #
```

## 14.1.3 /etc/sysconfig/network/dhcp

В данном файле задаются настройки DHCP клиента, используемого в системе. Каждая опция данного файла снабжена подробными комментариями про ее предназначение. В частности, опция *DHCLIENT\_BIN* задает имя dhcp клиент, который используется для инициализации сетевых интерфейсов. В частности, возможны варианты:

*DHCLIENT\_BIN=“dhpcd”*  
*DHCLIENT\_BIN=“dhclient”*

Если данная опция остается незаполненной, то сначала используется клиент по имени *dhpcd*, а затем *dhclient*.

Для логирования информации, связанной с работой DHCP клиента, необходимо опции *DHCLIENT\_DEBUG* присвоить значение “yes”. Эта опция может быть полезной для анализа работы DHCP протокола в вашей сети.





## 14.1.4 /etc/sysconfig/network/routes

В данном файле задаются общие для всех интерфейсов сетевые маршруты. Информация о маршрутах задается в следующем виде:

Адрес_назначения	Шлюз	Сетевая_маска	Устройство	Тип_Маршрута
Поле	Описание			
Адрес назначения	IP адрес сети или узла. Можно использовать и FQDN, если доступен DNS сервер.			
Шлюз	Шлюз по умолчанию или шлюз, через который узлы или сети могут быть доступны.			
Сетевая маска	Сетевая маска сети или узла за шлюзом. Например, 255.255.255.255 – маска для узла.			
Устройство	Имя интерфейса, используемого для отправки пакетов. Например, eth0.			
Тип маршрута	Опциональное поле, подробнее о котором можно узнать командой <i>man 5 route</i>			

Если в какой-либо строке нет необходимости в заполнении одного из выше указанных полей, то на его месте необходимо указать символ «-».

## 14.1.5 /etc/resolv.conf

В данном файле указывается список используемых DNS серверов (опция *nameserver*), а также доменные суффиксы подключений (опция *search*).

Когда DNS клиент пытается разрешить имя, указанное не в FQDN формате, он последовательно подставляет суффиксы, указанные в опции *search* к указанному имени. Таким образом, можно указывать несколько суффиксов. Можно указывать и несколько DNS серверов, каждый из которых должен быть указан в новой строке и начинаться с опции *nameserver*. Например:

```
search itstep.org
nameserver 8.8.8.8
nameserver 10.0.0.11
```

Содержимое данного файла может изменяться автоматически, с использованием скрипта *netconfig*. В частности, опция *NETCONFIG\_DNS\_STATIC\_SEARCHLIST* файла */etc/sysconfig/network/config* определяет опции *search* файла */etc/resolv.conf*. А опция



`NETCONFIG_DNS_STATIC_SERVERS` – содержимое опций *nameserver*. Для отключения DNS конфигурирования с использованием *netconfig*, необходимо установить пустое значение опции `NETCONFIG_DNS_POLICY` в файле `/etc/sysconfig/network/config`.

### 14.1.6 /sbin/netconfig

Данная утилита состоит из ряда модулей, каждый из которых может изменять сетевую конфигурацию системы, записывая информацию в соответствующие конфигурационные файлы и перезапуская службы. Статические настройки и политики, определяющие работу *netconfig*, указаны в файле `/etc/sysconfig/network/config`. Переменные данного файла могут быть инициализированы путем редактирования файла, а также с использованием YaST.

Для пользователей доступно лишь одно действие данной утилиты: *netconfig update*. Этой командой производится обновление сетевых настроек на основании текущих настроек. Такая команда может быть полезна для применения настроек после изменения конфигурационных файлов.

Для служб и утилит доступны дополнительные действия: *netconfig modify* и *netconfig remove*, с помощью которых конфигурационные файлы сетевых настроек системы могут быть подвержены редактированию различными модулями утилиты *netconfig*. Например, модуль *dns-resolver* имеет возможность редактировать содержимое файла `/etc/resolv.conf`.

### 14.1.7 /etc/hosts

В данном файле производится сопоставление IP адреса каких-либо узлов с их доменными именами. В случае, если в нашей сетевой инфраструктуре отсутствует DNS сервер, доступ к узлам до доменным именам может быть организован с помощью файла `/etc/hosts`.

Для каждого узла в новой строке нужно указать IP адрес и соответствующее ему FQDN имя. В качестве разделителей можно использовать пробелы и символы табуляции. Например:

`127.0.0.1 localhost`  
`8.8.8.8 google-public-dns-a.google.com`





## 14.1.8 /etc/host.conf

Данный файл контролирует поведение локального DNS resolver-а. В частности, здесь могут быть указаны следующие параметры:

Поле	Описание
<i>Order host, bind</i>	Задает порядок разрешения имен.  <i>hosts</i> – поиск в файле /etc/hosts. <i>bind</i> – обращение к DNS серверу.
<i>Multi on/off</i>	Определяет, может ли узел, указанный в файле /etc/hosts, иметь несколько IP адресов

Пример:

```
order hosts bind
multi on
```

## 14.1.9 /etc/HOSTNAME

В данном файле должна находиться только одна строка, в которой указано символьное имя системы. Некоторые службы и утилиты обращаются к переменной *\$HOSTNAME* или к содержимому данного файла, для получения информации об имени узла. Поэтому, заполнение данного файла является важной частью сетевых настроек.

## 14.2 Тестирование сетевой конфигурации

Прежде чем заполнять конфигурационные файлы, следует протестировать предполагаемые настройки. Для этого следует использовать утилиту *ip*. Она заменяет собой устаревшие утилиты *ifconfig* и *route*, которые, впрочем, тоже можно использовать.

Команды *ip*, *route*, *ifconfig* изменяют сетевые настройки без сохранения информации в конфигурационные файлы. Поэтому, любые изменения, которые вносятся ими в системе, будут потеряны при перезагрузке.



## 14.2.1 ip

Утилита `ip` многогранна, с помощью нее можно просматривать и изменять конфигурацию устройств, редактировать таблицу маршрутизации, создавать туннели.

Формат команды:

`ip опции объект команда`

Работать можно со следующими объектами:

Объект	Описание объекта
<code>Link</code>	Сетевое устройство
<code>Address</code>	IP адрес сетевого устройства
<code>Neighbour</code>	Запись в ARP-cache
<code>Route</code>	Запись в таблице маршрутизации
<code>Rule</code>	Правило в базе данных правил маршрутизации
<code>Maddress</code>	Multicast адрес
<code>Mroute</code>	Multicast кэш маршрутизации
<code>Tunnel</code>	IP туннель

Если не задана какая-либо команда, то используется команда по умолчанию – `list`. Для изменения состояния сетевого устройства используется команда `ip link set имя_устройства команда`. Например, команды выключения и включения интерфейса `eth0`:

```
ip Link set eth0 down
ip link set eth0 up
```

После того, как устройство активировано, его можно настраивать. Для установки IP адреса используется следующий синтаксис: `ip addr add ip_адрес + dev имя_устройства`. Например, для установки адреса интерфейса `eth0` в `192.168.12.154/30` со стандартным широковещательным адресом (опция `brd`), необходимо использовать команду:





```
ip addr add 192.168.12.154/30 brd + dev eth0.
```

Для работы понадобится и шлюз по умолчанию. Для установки его в, используется следующая конструкция:

```
ip route add gateway_ip_address
```

Для отображения всех устройств используется следующая команда:

```
ip link ls
```

Для отображения только активированных интерфейсов, следующая команда:

```
ip link ls up
```

Для вывода статистики по интерфейсу, можно воспользоваться командой:

```
ip -s link ls device_name
```

Для просмотра IP и MAC адресов интерфейсов:

```
ip addr
```

Для просмотра таблицы маршрутизации:

```
ip route show
```

Для получения справки необходимо использовать опцию *help*, которую можно указывать для всех объектов. Например, помощь по команде *ip* и по команде *ip addr*:

```
ip help
ip addr help
```

### 14.3 Задания для самопроверки

1. Настройте сетевые интерфейсы вашей системы таким образом:  
eth0 – статический IP адрес в приватной сети.  
eth1 – адрес, получаемый по DHCP от сервера вашей организации.  
Убедитесь в том, что имеется доступ к узлам в сети Internet.



## 15. DHCP-сервер Dhcpd

Программное обеспечение, используемое в качестве DHCP сервера и DHCP клиента, доступно в любом современном Linux дистрибутиве. Для создания DHCP сервера используется, как правило, *dhcpd* сервер, который разрабатывает Internet Systems Consortium (*ISC*).

В качестве клиентского программного обеспечения используется, как правило, один из двух вариантов. Это клиентская служба *dhcpcd* и клиент под названием *dhclient*, который поставляется в пакете *dhcp-client* и разрабатывается также *ISC*. Клиент *dhcpcd* проще в использовании, так как не требует какой-либо конфигурации. Если необходима тонкая настройка клиента, предпочтительнее использовать *dhclient*, конфигурация которого задается в файле */etc/dhclient.conf*

### 15.1 dhcpcd

Данный сервер поставляется в пакете под названием *dhcp-server*. Запуск, остановку и другие действия над данной службой можно осуществлять с помощью сценария */etc/init.d/dhcpcd*. Для настройки *dhcpcd* следует редактировать конфигурационные файлы */etc/dhcpcd.conf* и */etc/sysconfig/dhcpcd*. Для проверки наличия ошибок в конфигурационных файлах используется команда:

```
/etc/init.d/dhcpcd check-syntax
```

После их настройки конфигурационных файлов следует запустить или перезапустить службу *dhcpcd*, если она уже была запущена.

```
/etc/init.d/dhcpcd restart
```

Чтобы служба *dhcpcd* автоматически запускалась при старте системы, необходимо воспользоваться утилитой *chkconfig*:

```
chkconfig dhcpcd on
```

После успешной настройки и запуска сервера, в файле */var/lib/dhcp/db/dhcpcd.leases* можно просматривать информацию об арендах, выданных сервером.



## 15.1.1 /etc/dhcpd.conf

Данный файл условно можно разбирать на 3 части. Первая часть – описание глобальных параметров для всех подсетей, обслуживаемых сервером. Вторая часть – описание подсетей и параметров, применяемых только к ним. Третья часть – описание параметров отдельных узлов сети, для которых должны применяться особые настройки.

Рассмотрим пример глобальных настроек. Для начала, зададим временные параметры.

```
default-lease-time 600;      # Через какое количество секунд клиент, получивший в аренду
                            # адрес, должен сделать запрос к серверу о продлении аренды

max-lease-time 7200;        # Какое максимальное количество секунд клиент может
                            # использовать адрес без продления аренды
```

Далее, определим сетевые настройки, общие для всех сетей, обслуживаемых DHCP сервером.

```
option domain-name "itstep.org";          # Имя домена, используемого в сети.

option domain-name-servers 192.168.1.11;    # Адреса DNS серверов сети.
                                              # Можно указать не более трех серверов.

option broadcast-address 192.168.2.255;     # Широковещательный адрес сети, который
                                              # должен использовать клиентский узел

option routers 192.168.2.1;                  # Адрес шлюза по умолчанию

option subnet-mask 255.255.255.0;            # Сетевая маска клиентских узлов сети

ddns-update-style none;                      # Не использовать динамическое обновление
                                              # DNS зоны. Обязательный параметр в ISC
                                              # dhcpd версии 3.
```

Следующий этап – описание сети:

```
subnet 192.168.2.0 netmask 255.255.255.0 {
    range 192.168.2.10 192.168.2.20;
    range 192.168.2.100 192.168.2.200;
}                                         # Узлы будут получать адреса в сети
                                              # 192.168.2.0/24 в пределах двух
                                              # диапазонов:
                                              # от 192.168.2.10 до 192.168.2.20
                                              # и от 192.168.2.100 до 192.168.2.200
```





Заключительный этап – описание настроек отдельных узлов, идентифицируются которые по своему MAC адресу.

```
host jupiter {  
    hardware ethernet 00:30:6E:08:EC:80; # За узлом с MAC адресом 00:30:6E:08:EC:80  
    fixed-address 192.168.2.100; # резервируется IP адрес 192.168.2.100  
}
```

Следует отметить, что опции, используемые в качестве глобальных, могут быть указаны и в описании конкретной сети или узла. В таком случае, они будут применяться только к клиентским узлам данной сети или к конкретным узлам. Например:

```
subnet 10.0.6.96 netmask 255.255.255.224 {  
    range 10.0.6.100 10.0.6.120; # Узлы будут получать адреса в сети  
    option routers 10.0.6.97; # 10.0.6.96/27 в пределах диапазона:  
    option domain-name "internal.itstep.org" # от 10.0.6.100 до 10.0.6.120  
    default-lease-time 7200; # шлюз по умолчанию – 10.0.6.97  
}  
# имя домена – internal.itstep.org  
# время аренды адреса – 2 часа
```

## 15.1.2 /etc/sysconfig/dhcpd

В данном файле могут настраиваться некоторые важные параметры работы сервера. Например, переменная *DHCPD\_INTERFACE* должна быть проинициализирована именем одного или более сетевых интерфейсов, на которых должен запускаться DHCP сервер. Например:

```
DHCPD_INTERFACE="eth0"  
DHCPD_INTERFACE="eth0 eth1"  
DHCPD_INTERFACE="ANY"
```

Значение “ANY” задает такое поведение сервера, при котором он будет автоматически определять любые доступные интерфейсы и запускаться на них.

Еще одна важная опция - *DHCPD\_RUN\_CHROOTED*. В современных серверных системах, некоторые сетевые службы могут быть запущены в виртуальном корневом каталоге. Это делается для усиления системы безопасности.

```
DHCPD_RUN_CHROOTED="yes"  
DHCPD_RUN_CHROOTED ="no"
```



Для DHCP сервера, запущенного в *chroot* окружении, корневым каталогом будет не каталог «/», а каталог */var/lib/dhcp*. Для нормальной работы службы, в каталоге */var/lib/dhcp* создаются дополнительные каталоги, в частности */var/lib/dhcp/etc*, в который автоматически копируются все необходимые для работы DHCP сервера конфигурационные файлы при запуске сценария */etc/init.d/dhcpd*. В частности, это такие известные нам файлы, как *dhcpd.conf*, *host.conf*, *hosts*, *resolv.conf*.

## 15.2 Задание для самопроверки

1. Настройте DHCP сервер, который обслуживает сеть 192.168.200.0/24 и раздает адреса в диапазоне 192.168.200.10 – 192.168.200.20.
2. Зарезервируйте за одним из узлов вашей сети адрес 192.168.200.200.
3. Используя Windows и Linux клиентов, проверьте функциональность сервера.
4. В список опций, передаваемых клиентам, добавьте адрес шлюза по умолчанию и адрес DNS сервера.
5. Обновите аренду адреса со стороны клиентов, проанализируйте полученные ими сетевые настройки.



## 16 DNS-сервер Bind

В качестве DNS сервера в современных Linux системах используется сервер под названием *named*, который входит в состав дистрибутива под названием BIND (*Berkley Internet Name Domain*). BIND, так же как и *dhcpd*, поставляется организацией под названием *ISC*.

Как правило, *named* устанавливается сконфигурированным так, что сразу после инсталляции системы локальную службу доменных имен уже можно использовать. Если в системе есть активное *Internet* соединение и в качестве *nameserver*-а в файле */etc/resolv.conf* указан *localhost* или *127.0.0.1*, то в такой системе уже работает служба доменных имен даже без знания адресов DNS серверов провайдера.

Дело в том, что *named* поставляется со списком корневых серверов (*root hints*), к которым он и обращается при попытке разрешения доменного имени узла.

Если есть возможность использовать DNS сервер провайдера, то разрешение имен можно сделать более эффективным, указав IP адрес данного сервера в опции *forwarders* файла */etc/named.conf*. Это позволит использовать локальный DNS сервер в качестве кэширующего (*caching-only*) сервера.

Для полноценной работы сервера, ему необходимо создать локальную зону, для которой он будет выступать в роли ответственного (*authoritative*) сервера.

Для запуска службы *named* используется стандартная команда:

```
/etc/init.d/named start
```

Для настройки сервера необходимо редактировать файлы */etc/named.conf* и */etc/sysconfig/named*. Файлы зон, которые обслуживаются сервером, хранятся, как правило, в каталоге */var/lib/named*.

После редактирования конфигурационных файлов и перед запуском сервера, есть смысл проверить правильность настроек, указанных в файле */etc/named.conf* и в файлах зон. Это, соответственно, выполняется следующими командами:

```
named-checkconf  
named-checkzone имя_зоны файл_зоны
```



## 16.1 /etc/named.conf

Конфигурационный файл `/etc/named.conf` условно можно разбить на две основные части. Первая – это глобальные настройки сервера, указанные в секции `options`. Вторая – это настройки отдельных доменов, каждый из которых указан в опции `zone`.

Рассмотрим пример простейшей конфигурации файла `/etc/named.conf`:

```
options {                                     # Начало секции глобальных настроек
    directory "/var/lib/named";             # Каталог, где хранятся файлы зон
    forwarders { 10.0.0.1; };               # Адрес DNS сервера, на который нужно
                                            # направлять запросы, если локальный сервер
                                            # не смог разрешить DNS имя
    notify no;                            # Не сообщать другим серверам о том, что
                                            # файл зоны был отредактирован или о том, что
                                            # сервер был перезапущен
};

# Конец секции глобальных настроек

zone "itstep.org" in {
    type master;                         # Начало описания зоны под именем "itstep.org"
    file "itstep.zone";                  # Тип зоны – "master" (основная)
    allow-update {! *};                 # Имя файла зоны – "itstep.zone"
                                        # (в каталоге /var/lib/named)
    # Запретить обновления зоны клиентами извне
    # (по умолчанию также установлен запрет)
    # Конец описания зоны "itstep.org"
};

zone "example.net" in {
    type slave;                          # Начало описания зоны под именем "example.net"
    file "slave/example.zone";          # Тип зоны – "slave" (подчиненная)
    masters { 192.168.0.1; };           # Имя файла зоны - "slave/example.zone"
                                        # Адрес основного сервера - 192.168.0.1
    # Конец описания зоны
};

zone "." in {
    type hint;                           # Начало описания корневой зоны под именем "."
    file "root.hint";                   # Тип зоны – "hint" (root hints)
    # Имя файла зоны – "root.hint"
    # Конец описания зоны
};
```

Помимо указанных в примере параметров секции глобальных настроек, в файле `/etc/named.conf` также могут быть использованы и такие директивы (полную информацию можно получить командой `man 5 named.conf`):





### Пример директивы

### Объяснение

*forward first;*

Прежде чем обращаться к корневым серверам, переадресовать запрос на сервер, указанный в опции *forwarders*.

*forward only;*

Аналогично предыдущему, но обращение к корневым серверам никогда не производится. Эта опция применима, когда правила брандмауэра позволяют обращаться только лишь на DNS сервер провайдера.

*listen-on port 53 {127.0.0.1; IP-адрес; };*

IP адрес и порт, на которых сервер принимает запросы клиентов. Номер порта 53 можно не указывать, так как это порт по умолчанию. Если не указывать данную опцию, то, по умолчанию, «прослушиваются» все интерфейсы

*query-source address \* port 53;*

Когда серверу необходимо переадресовать запрос, он отсылает его с собственного 53 порта. Опция применима, когда правила брандмауэра ограничивают использование портов.

*allow-query {127.0.0.1; адрес\_сети; };*

Определяет сети, из которых могут поступать запросу на сервер. Пример сети: 192.168.10.0/24

*allow-transfer {! \*; };*

Определяет, какие узлы могут запрашивать передачу зоны у сервера. Запись “! \*” полностью запрещает какие-либо передачи зон. По умолчанию передача зоны разрешена на любые узлы.

*statistics-interval 0;*

BIND генерирует несколько строк статистической информации в файле */var/log/messages* один раз в час. Можно установить альтернативное значение обновления в минутах. Значение 0 – отключает это поведение сервера.

*cleaning-interval 720;*

Период времени, по окончании которого BIND очищает DNS кэш. Время указывается в минутах. По умолчанию – 60.

*interface-interval 0;*

BIND периодически производит поиск сетевых интерфейсов, которые появились в системе или исчезли. Если установлено значение 0, такая проверка не производится и сервер работает только на тех интерфейсах, которые были активны при старте службы *named*. Интервал указывается в минутах. По умолчанию – 60.





## 16.2 Файлы зон

Для полноценного обслуживания домена, для него должно быть созданы два файла зоны. Первый файл – с прямой зоной, второй файл – с обратной зоной. Прямая зона служит для преобразования доменных имен в IP адреса, а обратная, соответственно, для обратного преобразования.

### 16.2.1 Прямая зона

Рассмотрим пример содержимого файла прямой зоны:

```
$TTL 2D ; Время жизни каждой записи в данном файле
; В примере - 2 дня (2D)

itstep.org. IN SOA ns root.itstep.org. ( ; SOA запись. Имя домена: itstep.org.
; имя ответственного сервера: ns.itstep.org.
; e-mail администратора домена: root@itstep.org.
2010112601 ; serial ; Серийный номер зоны
1D ; refresh ; Интервал обновления зоны подчиненным сервером
2H ; retry ; Интервал повторной попытки обновления зоны
1W ; expiry ; Срок жизни зоны, которую не удалось обновить
2D) ; minimum ; Срок жизни кэшированных негативных ответов

IN NS ns ; NS запись - задает имя DNS сервера
IN MX 10 mail ; MX запись - задает имя mail сервера
; и его приоритет (чем меньше, тем выше)

ns ; A запись - IP адрес узла ns.itstep.org
gate ; A запись - IP адрес узла gate.itstep.org
mail ; A запись - IP адрес узла mail.itstep.org
jupiter ; A запись - IP адрес узла jupiter.itstep.org
www ; A запись - IP адрес узла www.itstep.org

ftp ; CNAME запись - псевдоним узла www.itstep.org
```

При указании доменных имен в файле зоны необходимо помнить о том, что имена должны заканчиваться символом «.». Если в зоне присутствует имя узла, которое не оканчивается символом «.», то к такому имени будет автоматически добавлен суффикс, соответствующий имени домена.

В выше приведенном примере в разных местах файла зоны указаны имена, в конце которых нет символа «.». Это узлы *gate*, *ns*, *mail* и т.д. Все такие имена автоматически преобразовываются в имена *gate.itstep.org*, *ns.itstep.org*, *mail.itstep.org*.





Для обозначения имени зоны в файле может быть использован символ «@». Так как этот символ имеет специальное назначение, в записи SOA e-mail администратора указан с разделителем «.». Это означает, что адрес вида *root@itstep.org* в SOA записи будет указан как *root.itstep.org*.

### 16.2.2 Обратная зона

Рассмотрим пример содержимого файла прямой зоны. Многие его параметры, лишь за некоторым исключением, аналогичны параметрам файла прямой зоны:

```
$TTL 2D ; Время жизни
168.192.in-addr.arpa. IN SOA ns.itstep.org. root.itstep.org. ( ; SOA запись. Используются
; обязательно полные имена
2010112601 ; serial ; Серийный номер зоны
1D ; refresh ; Интервал обновления зоны
2H ; retry ; Интервал повтора
1W ; expiry ; Срок жизни зоны
2D) ; minimum ; TTL кэшированных данных

IN NS ns.itstep.org. ; NS запись

1.11 IN PTR ns.itstep.org. ; PTR запись – имя узла
; под адресом 192.168.1.11
1.1  IN PTR gate.itstep.org. ; имя узла 192.168.1.1
1.12  IN PTR mail.itstep.org. ; имя узла 192.168.1.12
2.100 IN PTR jupiter.itstep.org. ; имя узла 192.168.2.100
1.2   IN PTR www.itstep.org. ; имя узла 192.168.1.2
```

После того, как созданы обе зоны и должным образом отредактирован файл /etc/named.conf, можно запускать сервер и тестировать его работу с помощью таких утилит, как *host*, *dig* и *nslookup*.

### 16.3 Задание для самопроверки

- Настройте основной DNS сервер, обслуживающий домен *test.org* в сети 192.168.200.0/24. Создайте прямую и обратную зону с записями об узлах вашей сети. Проверьте работоспособность сервера с помощью утилит *host*, *dig* и *nslookup*.
- Настройте подчиненный сервер, обслуживающий зону *test.org* и получающий файл зоны с основного сервера, настроенного вами в первом упражнении.



## 17 Брандмауэр Netfilter и утилита Iptables

*Netfilter* – это брандмауэр (межсетевой экран), встроенный в ядро Linux версий 2.4 и 2.6. *Netfilter* состоит из нескольких модулей ядра, которые используются для разных протоколов. Для управления работы брандмауэра используются различные утилиты. Например, утилита командной строки *iptables* используется для создания и удаления правил, применяющихся к протоколу IPv4, *ip6tables* – для протокола IPv6, *arptables* – для протокола ARP, *ebtables* – для протокола Ethernet. *Netfilter* состоит из нескольких таблиц, каждая таблица из нескольких цепочек, по которым следует сетевой пакет, попавший на брандмауэр. В каждой цепочке – список правил, содержащих в себе критерии, определяющие, попадает ли пакет под заданное правило, и действие, которое необходимо выполнить в случае выполнения критерия.

В общем виде правила записываются так:

```
iptables [-t table] command [match] [target/jump]
```

Если в правило не включается спецификатор *[-t table]*, то по умолчанию предполагается использование таблицы filter, если же предполагается использование другой таблицы, то это требуется указать явно. Спецификатор таблицы так же можно указывать в любом месте строки правила, однако более или менее стандартом считается указание таблицы в начале правила.

Далее, непосредственно за именем таблицы, должна стоять команда. Если спецификатора таблицы нет, то команда всегда должна стоять первой. Команда определяет действие *iptables*, например: вставить правило, или добавить правило в конец цепочки, или удалить правило и т.п.

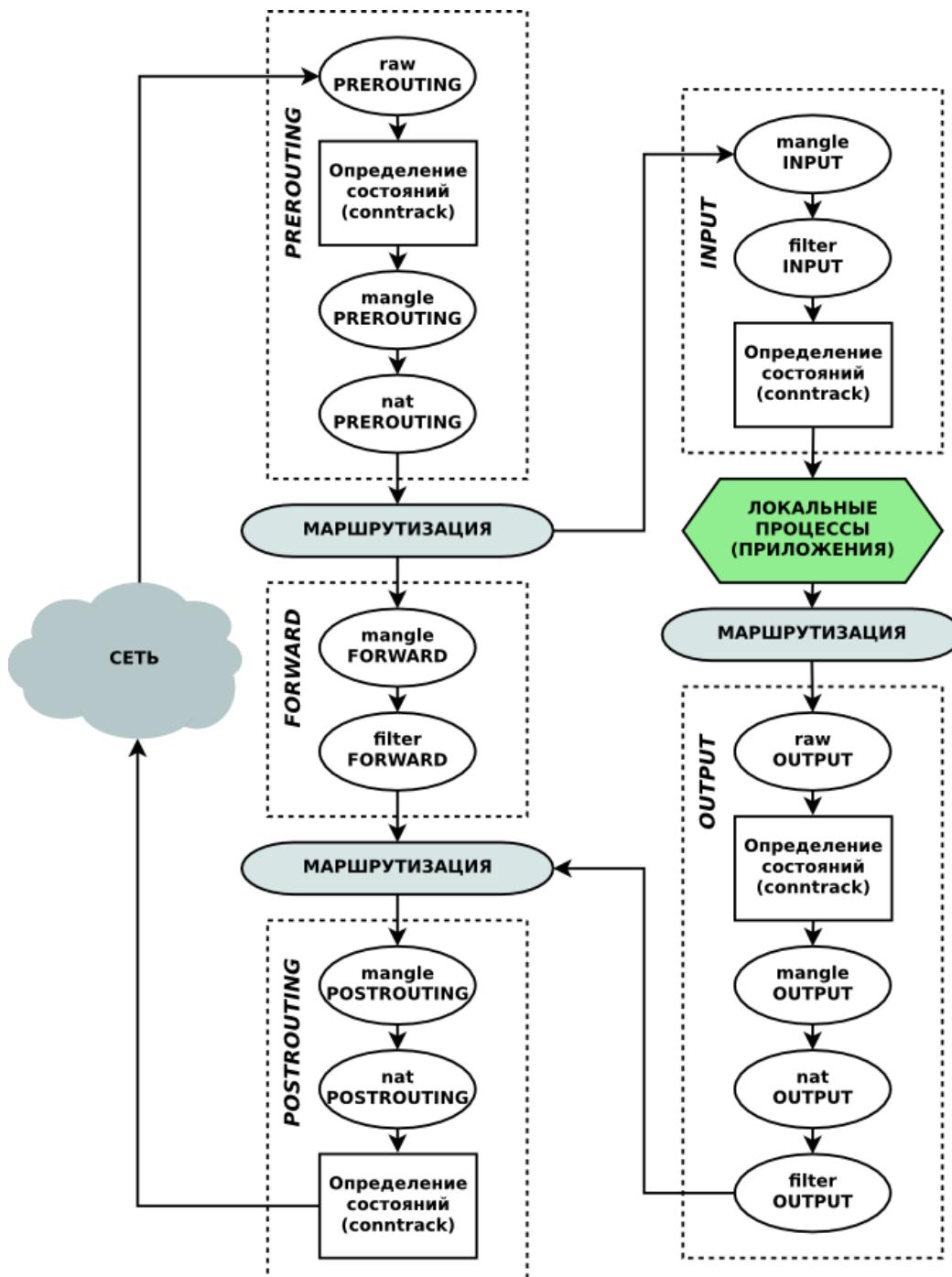
Раздел *matches* задает критерии проверки, по которым определяется, подпадает ли пакет под действие этого правила или нет. Здесь мы можем указать самые разные критерии – и IP-адрес источника пакета или сети, и сетевой интерфейс и т.д. Существует множество критериев, которые мы рассмотрим в данной главе.

И наконец, *target* указывает, какое действие должно быть выполнено при условии выполнения критериев в правиле. Здесь можно заставить ядро передать пакет в другую цепочку правил, "бросить" пакет и забыть про него, выдать на источник сообщение об ошибке и т.п.



## 17.1 Порядок движения пакетов

Процесс движения пакетов иллюстрирует диаграмма:



В зависимости от того, является ли пакет транзитным или локальным, он обрабатывается различными цепочками таблиц Netfilter.



## 17.1.1 Порядок движения транзитных пакетов

ШАГ	Таблица	Цепочка	Описание
1			Кабель (интернет)
2			Сетевой интерфейс (например, eth0)
3	mangle	PREROUTING	Обычно эта цепочка используется для внесения изменений в пакет, например для изменения битов TOS и TTL.
4	nat	PREROUTING	Эта цепочка используется для трансляции сетевых адресов (Destination Network Address Translation). Source Network Address Translation выполняется позднее, в другой цепочке. Любой рода фильтрация в этой цепочке может производиться только в исключительных случаях.
5			Принятие решения о дальнейшей маршрутизации, т.е. в этой точке решается, куда пойдет пакет - локальному приложению или на другой узел сети.
6	mangle	FORWARD	Используется только в исключительных случаях, когда необходимо внести некоторые изменения в заголовок пакета между двумя точками принятия решения о маршрутизации.
7	filter	FORWARD	В цепочку FORWARD попадают только те пакеты, которые идут на другой сетевой узел. Вся фильтрация транзитного трафика должна выполняться здесь. Не забывайте, что через эту цепочку проходит сетевой трафик в обоих направлениях, обязательно учитывайте это обстоятельство при написании правил фильтрации.
8	mangle	POSTROUTING	Эта цепочка предназначена для внесения изменений в заголовок пакета уже после того как принято последнее решение о маршрутизации.
9	nat	POSTROUTING	Эта цепочка предназначена в первую очередь для Source Network Address Translation. Не используйте ее для фильтрации без особой необходимости. Здесь же выполняется маскарадинг (Masquerading).
10			Выходной сетевой интерфейс (например, eth1)
11			Кабель (например, подключенный к локальной внутренней сети)

Итак, пакет проходит несколько этапов, прежде чем он будет передан далее. На каждом из них пакет может быть остановлен. Цепочку FORWARD проходят ВСЕ пакеты, которые движутся через наш брандмауэр/роутер. Необходимо помнить, что транзитные пакеты никогда не попадают в цепочку INPUT.



## 17.1.2 Порядок движения пакетов для локальных процессов

ШАГ	Таблица	Цепочка	Описание
1			Кабель (интернет)
2			Сетевой интерфейс (например, eth0)
3	mangle	PREROUTING	Внесение изменений в пакет, например изменение битов TOS и TTL.
4	nat	PREROUTING	Трансляции сетевых адресов (Destination Network Address Translation).
5			Принятие решения о дальнейшей маршрутизации.
6	mangle	INPUT	Здесь вносятся изменения в заголовок пакета, прежде чем он будет передан локальному приложению.
7	filter	INPUT	Здесь производится фильтрация входящего трафика. Все входящие пакеты, адресованные нам, проходят через эту цепочку, независимо от того, с какого интерфейса они поступили.
8			Локальный процесс или приложение

Итак, в данном случае, все пакеты проходят через цепочку INPUT, а не FORWARD.

## 17.1.3 Порядок движения пакетов от локальных процессов

ШАГ	Таблица	Цепочка	Описание
1			Локальный процесс (программа – сервер или программа – клиент).
2			Принятие решения о маршрутизации.
3	mangle	OUTPUT	Внесение изменений в заголовок пакета.
4	nat	OUTPUT	Трансляция адресов в пакетах, исходящих от локальных процессов.
5	filter	OUTPUT	Фильтрация исходящего трафика.
6	mangle	POSTROUTING	Модификация заголовков пакета, перед тем как он покинет брандмауэр.
7	nat	POSTROUTING	Выполняется SNAT.
8			Сетевой интерфейс (например, eth0).
9			Кабель (Интернет).



## 17.2 Команды iptables

При формировании правил обработки пакетов, указываются команды, посредством которых мы указываем, следует ли новое правило добавить в конец или в начало цепочки, нужно ли его удалить и т.д. Команда должна быть указана всегда. Список доступных команд можно просмотреть с помощью команды *iptables -h* или, что то же самое, *iptables --help*. Некоторые команды могут использоваться совместно с дополнительными ключами. Далее рассмотрены команды, которые используются в *iptables*.

### -A, --append

Пример: *iptables -A INPUT ...*

Добавляет новое правило в конец заданной цепочки.

### -D, --delete

Пример: *iptables -D INPUT --dport 80 -j DROP, iptables -D INPUT 1*

Удаление правила из цепочки. Команда имеет два формата записи, первый -- когда задается критерий сравнения с опцией -D (см. первый пример), второй -- порядковый номер правила. Если задается критерий сравнения, то удаляется правило, которое имеет в себе этот критерий, если задается номер правила, то будет удалено правило с заданным номером. Счет правил в цепочках начинается с 1.

### -R, --replace

Пример: *iptables -R INPUT 1 -s 192.168.0.1 -j DROP*

Данная команда заменяет одно правило другим. В основном она используется во время отладки новых правил.

### -I, --insert

Пример: *iptables -I INPUT 1 --dport 80 -j ACCEPT*

Вставляет новое правило в цепочку. Число, следующее за именем цепочки, указывает номер правила, перед которым нужно вставить новое правило, другими словами число задает номер для вставляемого правила. В примере выше, указывается, что данное правило должно быть 1-м в цепочке INPUT.



**-L, --list**

Пример: *iptables -L INPUT*

Вывод списка правил в заданной цепочке, в данном примере предполагается вывод правил из цепочки INPUT. Если имя цепочки не указывается, то выводится список правил для всех цепочек. Формат вывода зависит от наличия дополнительных ключей в команде, например *-n*, *-v*, и пр.

**-F, --flush**

Пример: *iptables -F INPUT*

Сброс (удаление) всех правил из заданной цепочки (таблицы). Если имя цепочки и таблицы не указывается, то удаляются все правила, во всех цепочках.

**-Z, --zero**

Пример: *iptables -Z INPUT*

Обнуление всех счетчиков в заданной цепочке. Если имя цепочки не указывается, то подразумеваются все цепочки. При использовании ключа *-v* совместно с командой *-L*, на вывод будут поданы и состояния счетчиков пакетов, попавших под действие каждого правила. Допускается совместное использование команд *-L* и *-Z*. В этом случае будет выдан сначала список правил со счетчиками, а затем произойдет обнуление счетчиков.

**-N, --new-chain**

Пример: *iptables -N allowed*

Создается новая цепочка с заданным именем в заданной таблице. В выше приведенном примере создается новая цепочка с именем allowed. Имя цепочки должно быть уникальным и не должно совпадать с зарезервированными именами цепочек и действий (DROP, REJECT и т.п.)

**-X, --delete-chain**

Пример: *iptables -X allowed*

Удаление заданной цепочки из заданной таблицы. Удаляемая цепочка не должна иметь правил и не должно быть ссылок из других цепочек на удаляемую цепочку. Если имя цепочки не указано, то будут удалены все цепочки, определенные командой *-N* в



заданной таблице.

#### **-P, --policy**

Пример: *iptables -P INPUT DROP*

Определяет политику по умолчанию для заданной цепочки. Политика по умолчанию определяет действие, применяемое к пакетам, не попавшим под действие ни одного из правил в цепочке. В качестве политики по умолчанию допускается использовать DROP, ACCEPT и REJECT.

#### **-E, --rename-chain**

Пример: *iptables -E allowed disallowed*

Команда -E выполняет переименование пользовательской цепочки. В примере цепочка allowed будет переименована в цепочку disallowed. Эти переименования не изменяют порядок работы, а носят только косметический характер.

## **17.3 Критерии iptables**

Каждое правило, которое находится в цепочке, выделяет пакет о определенным критериям. Данные критерии можно разбить на пять групп. Первая группа – общие критерии, которые могут применяться к любым пакетам. Вторая и третья – критерии, применяющиеся только TCP и UDP пакетам соответственно. Четвертая – ICMP критерии, пятая – специальные критерии, такие как state, owner, limit и другие. Рассмотрим каждую категорию в отдельности.

### **17.3.1 Общие критерии**

Общие критерии допустимо употреблять в любых правилах и не зависят от типа протокола и не требуют подгрузки модулей расширения. В эту группу можно добавить и критерий *--protocol* несмотря на то, что он используется в некоторых специфичных от протокола расширениях.

Например, мы решили использовать TCP критерий, тогда нам необходимо будет использовать и критерий *--protocol* которому в качестве дополнительного ключа передается название протокола *--tcp*. Однако *--protocol* сам по себе является критерием, который используется для указания типа протокола.

■ Киев	■ Днепропетровск	■ Львов	■ Ровно	■ Мариуполь	■ Полтава
■ Одесса	■ Донецк	■ Николаев	■ Запорожье	■ Луганск	■ Харьков



**-p, --protocol**

Пример: *iptables -A INPUT -p tcp*

Этот критерий используется для указания типа протокола. Примерами протоколов могут быть TCP, UDP и ICMP. Прежде всего, в качестве имени протокола в данный критерий можно передавать три вышеупомянутых протокола, а также ключевое слово ALL. В качестве протокола допускается передавать число - номер протокола, так например, 255 соответствует протоколу RAW IP. Соответствия между номерами протоколов и их именами вы можете посмотреть в файле */etc/services*. Критерию может передаваться и список протоколов, разделенных запятыми, например так: *udp,tcp*. Если данному критерию передается числовое значение 0, то это эквивалентно использованию спецификатора ALL, который подразумевается по умолчанию, когда критерий *--protocol* не используется. Для логической инверсии критерия, перед именем протокола (списком протоколов) используется символ «!», например *--protocol ! tcp* подразумевает пакеты любого протокола, кроме *tcp*.

**-s, --src, --source**

Пример: *iptables -A INPUT -s 192.168.1.1*

IP-адрес(а) источника пакета. Адрес источника может указываться так, как показано в примере, тогда подразумевается единственный IP-адрес. А можно указать адрес в виде address/mask, например как 192.168.0.0/255.255.255.0, или более современным способом 192.168.0.0/24, т.е. фактически определяя диапазон адресов. Как и ранее, символ «!», установленный перед адресом, означает логическое отрицание, т.е. *--source ! 192.168.0.0/24* означает любой адрес кроме адресов 192.168.0.x

**-d, --dst, --destination**

Пример: *iptables -A INPUT -d 192.168.1.1*

IP-адрес(а) получателя. Имеет синтаксис схожий с критерием *--source*, за исключением того, что подразумевает адрес места назначения. Точно так же может определять как единственный IP-адрес, так и диапазон адресов. Символ «!» используется для логической инверсии критерия.

**-i, --in-interface**

Пример: *iptables -A INPUT -i eth0*



Интерфейс, с которого был получен пакет. Использование этого критерия допускается только в цепочках INPUT, FORWARD и PREROUTING, в любых других случаях будет вызывать сообщение об ошибке. При отсутствии этого критерия предполагается любой интерфейс, что равносильно использованию критерия *-i +*. Как и прежде, символ «!» инвертирует результат совпадения. Если имя интерфейса завершается символом «+», то критерий задает все интерфейсы, начинающиеся с заданной строки, например *-i PPP+* обозначает любой PPP интерфейс, а запись *-i ! eth+* - любой интерфейс, кроме любого eth.

### **-o, --out-interface**

Пример: *iptables -A FORWARD -o eth0*

Задает имя выходного интерфейса. Этот критерий допускается использовать только в цепочках OUTPUT, FORWARD и POSTROUTING, в противном случае будет генерироваться сообщение об ошибке. При отсутствии этого критерия предполагается любой интерфейс, что равносильно использованию критерия *-o +*. Как и прежде, символ «!» инвертирует результат совпадения. Если имя интерфейса завершается символом «+», то критерий задает все интерфейсы, начинающиеся с заданной строки, например *-o eth+* обозначает любой eth интерфейс, а запись *-o ! eth+* - любой интерфейс, кроме любого eth.

### **-f, --fragment**

Пример: *iptables -A INPUT -f*

Правило распространяется на все фрагменты фрагментированного пакета, кроме первого, сделано это потому, что нет возможности определить исходящий/входящий порт для фрагмента пакета, а для ICMP-пакетов определить их тип. С помощью фрагментированных пакетов могут производиться атаки на ваш брандмауэр, так как фрагменты пакетов могут не отлавливаться другими правилами. Как и раньше, допускается использования символа «!» для инверсии результата сравнения. только в данном случае символ «!» должен предшествовать критерию *-f*, например *! -f*. Инверсия критерия трактуется как "все первые фрагменты фрагментированных пакетов и/или нефрагментированные пакеты, но не вторые и последующие фрагменты фрагментированных пакетов".



### 17.3.2 Неявные критерии

В этом разделе мы рассмотрим неявные критерии, точнее, те критерии, которые подгружаются неявно и становятся доступны, например, при указании критерия `--protocol`. На сегодняшний день существует три автоматически подгружаемых расширения, это TCP критерии, UDP критерии и ICMP критерии. Загрузка этих расширений может производиться и явным образом с помощью ключа `-m`, `--match`, например `-m tcp`.

#### 17.3.2.1 TCP критерии

Это расширение зависит от типа протокола и работает только с TCP пакетами. Чтобы использовать эти дополнительные критерии, вам потребуется в правилах указывать тип протокола `--protocol tcp`. Важно: критерий `--protocol tcp` обязательно должен стоять перед специфичным критерием. Эти расширения загружаются автоматически как для `tcp` протокола, так и для `udp` и `icmp` протоколов.

##### **--sport, --source-port**

Пример: `iptables -A INPUT -p tcp --sport 22`

Исходный порт, с которого был отправлен пакет. В качестве параметра может указываться номер порта или название сетевой службы. Соответствие имен сервисов и номеров портов вы сможете найти в файле `/etc/services`. При указании номеров портов правила отрабатывают несколько быстрее, однако это менее удобно при разборе листингов скриптов. Если же вы собираетесь создавать значительные по объему наборы правил, скажем, порядка нескольких сотен и более, то тут предпочтительнее использовать номера портов. Номера портов могут задаваться в виде интервала из минимального и максимального номеров, например: `--source-port 22:80`. Если опускается минимальный порт, т.е. когда критерий записывается как `--source-port :80`, то в качестве начала диапазона принимается число 0. Если опускается максимальный порт, т.е. когда критерий записывается как `--source-port 22:`, то в качестве конца диапазона принимается число 65535. Допускается такая запись `--source-port 80:22`, в этом случае `iptables` поменяет числа 22 и 80 местами, т.е. подобного рода запись будет преобразована в `--source-port 22:80`. Как и раньше, символ `!` используется для инверсии. Так критерий `--source-port ! 22` подразумевает любой порт, кроме 22. Инверсия может применяться и к диапазону портов, например `--source-port ! 22:80`.





### --dport, --destination-port

Пример: *iptables -A INPUT -p tcp --dport 22*

Порт, на который адресован пакет. Аргументы задаются в том же формате, что и для --source-port.

### --tcp-flags

Пример: *iptables -p tcp --tcp-flags SYN,ACK,FIN SYN*

Определяет маску и флаги tcp-пакета. Пакет считается удовлетворяющим критерию, если из перечисленных флагов в первом списке в единичное состояние установлены флаги из второго списка. Так для вышеуказанного примера под критерий подпадают пакеты, у которых флаг SYN установлен, а флаги FIN и ACK сброшены. В качестве аргументов критерия могут выступать флаги SYN, ACK, FIN, RST, URG, PSH, а так же зарезервированные идентификаторы ALL и NONE. ALL -- значит ВСЕ флаги и NONE - НИ ОДИН флаг. Так, критерий --tcp-flags ALL NONE означает, что все флаги в пакете должны быть сброшены. Как и ранее, символ ! означает инверсию критерия Важно: имена флагов в каждом списке должны разделяться запятыми, пробелы служат для разделения списков.

### --syn

Пример: *iptables -p tcp --syn*

Критерий --syn является, по сути, реликтом, перекочевавшим из *ipchains*. Критерию соответствуют пакеты с установленным флагом SYN и сброшенными флагами ACK и FIN. Этот критерий аналогичен критерию --tcp-flags SYN,ACK,FIN SYN. Такие пакеты используются для открытия соединения TCP. Заблокировав такие пакеты, вы надежно заблокируете все входящие запросы на соединение, однако этот критерий не способен заблокировать исходящие запросы на соединение. Как и ранее, допускается инвертирование критерия символом «!». Так критерий ! --syn означает все пакеты, не являющиеся запросом на соединение, т.е. все пакеты с установленными флагами FIN или ACK.

### --tcp-option

Пример: *iptables -p tcp --tcp-option 16*

Критерий проверки опций TCP.



### 17.3.2.2 UDP критерии

В данном разделе будут рассматриваться критерии, специфичные только для протокола UDP. Эти расширения подгружаются автоматически при указании типа протокола *--protocol udp*. Важно отметить, что пакеты UDP не ориентированы на установленное соединение, и поэтому не имеют различных флагов, которые дают возможность судить о предназначении дейтаграммы. Получение UDP пакетов не требует какого-либо подтверждения со стороны получателя. Если они потеряны, то они просто потеряны (не вызывая передачу ICMP сообщения об ошибке). Это предполагает наличие значительно меньшего числа дополнительных критериев, в отличие от TCP пакетов.

#### **--sport, --source-port**

Команда: *iptables -A INPUT -p udp --sport 53*

Исходный порт, с которого был отправлен пакет. Формат аргументов полностью аналогичен критерию *--sport* для протокола TCP.

#### **--dport, --destination-port**

Пример: *iptables -A INPUT -p udp --dport 53*

Порт, на который адресован пакет. Формат аргументов полностью аналогичен в критерию *--source-port*.

### 17.3.2.3 ICMP критерии

Этот протокол используется, как правило, для передачи сообщений об ошибках и для управления соединением. Он не является подчиненным IP протоколу, но тесно с ним взаимодействует, поскольку помогает обрабатывать ошибочные ситуации. Заголовки ICMP пакетов очень похожи на IP заголовки, но имеют и отличия. Главное свойство этого протокола заключается в типе заголовка, который содержит информацию о том, что это за пакет. Например, когда мы пытаемся соединиться с недоступным хостом, то мы получим в ответ сообщение ICMP host unreachable. Существует только один специфичный критерий для ICMP пакетов. Это расширение загружается автоматически, когда мы указываем критерий *--protocol icmp*. Заметьте, что для проверки ICMP пакетов могут употребляться и общие критерии, поскольку известны и адрес источника и адрес назначения и пр.



### --icmp-type

Пример: *iptables -A INPUT -p icmp --icmp-type 8*

Тип сообщения ICMP Тип сообщения ICMP определяется номером или именем. Числовые значения определяются в RFC 792. Чтобы получить список имен ICMP значений выполните команду *iptables --protocol icmp -help*. Как и ранее, символ ! инвертирует критерий, например *--icmp-type ! 8*.

### 17.3.3 Явные критерии

Перед использованием этих расширений, они должны быть загружены явно, с помощью ключа *-m* или *--match*. Так, например, если мы собираемся использовать критерии *state*, то мы должны явно указать это в строке правила: *-m state* левее используемого критерия. Некоторые из этих критериев могут находиться в стадии разработки, и работать не всегда, однако, в большинстве случаев, они работают вполне устойчиво. Все отличие между явными и неявными критериями заключается только в том, что первые нужно подгружать явно, а вторые подгружаются автоматически.

#### --mac-source

Пример: *iptables -A INPUT --mac-source 00:00:00:00:00:01*

MAC адрес сетевого узла, передавшего пакет. MAC адрес должен указываться в форме XX:XX:XX:XX:XX:XX. Как и ранее, символ ! используется для инверсии критерия, например *--mac-source ! 00:00:00:00:00:01*, что означает - пакет с любого узла, кроме узла, который имеет MAC адрес 00:00:00:00:00:01. Этот критерий имеет смысл только в цепочках PREROUTING, FORWARD и INPUT и нигде более.

#### --limit

Пример: *iptables -A INPUT -m limit --limit 3/hour*

Устанавливается максимальное количество пакетов, которое правило пропустит за единицу времени. В качестве аргумента указывается число пакетов и время. Допустимыми считаются следующие единицы измерения времени: */second /minute /hour /day*. По умолчанию принято значение 3 пакета в час, или *3/hour*. Критерий *limit* должен подгружаться явно ключом *-m limit*. Прекрасно подходит для правил, производящих запись в системный журнал (*logging*) и т.п. Добавляя этот критерий, мы



тем самым устанавливаем предельное число пакетов в единицу времени, которое способно пропустить правило. Можно использовать символ ! для инверсии, например *-m ! limit*. В этом случае подразумевается, что пакеты будут проходить правило только после превышения ограничения.

### Расширение Multiport

Расширение *multiport* позволяет указывать в тексте правила несколько портов и диапазонов портов.

#### --source-port

Пример: *iptables -A INPUT -p tcp -m multiport --source-port 22,53,80,110*

Служит для указания списка исходящих портов. С помощью данного критерия можно указать до 15 различных портов. Названия портов в списке должны отделяться друг от друга запятыми, пробелы в списке не допустимы. Данное расширение может использоваться только совместно с критериями *the -p tcp* или *-p udp*. Главным образом используется как расширенная версия обычного критерия *--source-port*.

#### --destination-port

Пример: *iptables -A INPUT -p tcp -m multiport --destination-port 22,53,80,110*

Служит для указания списка входных портов. Формат задания аргументов полностью аналогичен *-m multiport --source-port*

#### --port

Пример: *iptables -A INPUT -p tcp -m multiport --port 22,53,80,110*

Данный критерий проверяет как исходящий, так и входящий порт пакета. Формат аргументов аналогичен критерию *--source-port* и *--destination-port*. Обратите внимание на то, что данный критерий проверяет порты обоих направлений, т.е. если вы пишите *-m multiport --port 80*, то под данный критерий подпадают пакеты, идущие с порта 80 на порт 80.

### Расширение owner

Расширение *owner* предназначено для проверки "владельца" пакета. Изначально данное расширение было написано как пример демонстрации возможностей *iptables*.





Допускается использовать этот критерий только в цепочке OUTPUT. Такое ограничение наложено потому, что на сегодняшний день нет реального механизма передачи информации о "владельце" по сети. Справедливости ради следует отметить, что для некоторых пакетов невозможно определить "владельца" в этой цепочке. К такого рода пакетам относятся различные ICMP responses. Поэтому не следует употреблять этот критерий к ICMP responses пакетам.

#### --uid-owner

Пример: *iptables -A OUTPUT -m owner --uid-owner 500*

Производится проверка "владельца" по User ID (UID). Подобного рода проверка может использоваться, к примеру, для блокировки выхода в Интернет отдельных пользователей.

#### --gid-owner

Пример: *iptables -A OUTPUT -m owner --gid-owner 0*

Производится проверка "владельца" пакета по Group ID (GID).

#### --pid-owner

Пример: *iptables -A OUTPUT -m owner --pid-owner 78*

Производится проверка "владельца" пакета по Process ID (PID). Этот критерий достаточно сложен в использовании, например, если мы хотим позволить передачу пакетов на HTTP порт только от заданного демона, то нам потребуется написать небольшой сценарий, который получает PID процесса (хотя бы через ps) и затем подставляет найденный PID в правила.

### Критерий state

Критерий state (признак состояния соединения) используется совместно с кодом трассировки соединений и позволяет нам получать информацию о трассировочном признаком состояния соединения, что позволяет судить о состоянии соединения, причем даже для таких протоколов как ICMP и UDP. Данное расширение необходимо загружать явно, с помощью ключа -m state.

#### --state

Пример: *iptables -A INPUT -m state --state RELATED,ESTABLISHED*

- |          |                  |            |             |             |           |
|----------|------------------|------------|-------------|-------------|-----------|
| ■ Киев   | ■ Днепропетровск | ■ Львов    | ■ Ровно     | ■ Мариуполь | ■ Полтава |
| ■ Одесса | ■ Донецк         | ■ Николаев | ■ Запорожье | ■ Луганск   | ■ Харьков |



Проверяется признак состояния соединения (state). На сегодняшний день можно указывать 4 состояния: INVALID, ESTABLISHED, NEW и RELATED. Рассмотрим их подробнее:

Состояние	Описание
NEW	Признак NEW сообщает о том, что пакет является первым для данного соединения. Это означает, что это первый пакет в данном соединении, который увидел модуль трассировщика. Например, если получен SYN пакет, являющийся первым пакетом для данного соединения, то он получит статус NEW. Однако пакет может и не быть SYN пакетом и тем не менее получить статус NEW. Это может породить определенные проблемы в отдельных случаях, но может оказаться и весьма полезным, например, когда желательно "подхватить" соединения, "потерянные" другими брандмауэрами или в случаях, когда таймаут соединения уже истек, но само соединение не было закрыто.
RELATED	Состояние RELATED одно из самых "хитрых". Соединение получает статус RELATED, если оно связано с другим соединением, имеющим признак ESTABLISHED. Это означает, что соединение получает признак RELATED тогда, когда оно инициировано из уже установленного соединения, имеющего признак ESTABLISHED. Хорошим примером соединения, которое может рассматриваться как RELATED, является соединение FTP-data, которое является связанным с портом FTP control, а так же DCC соединение, запущенное из IRC. Обратите внимание на то, что большинство протоколов TCP и некоторые из протоколов UDP весьма сложны и передают информацию о соединении через область данных TCP или UDP пакетов и поэтому требуют наличия специальных вспомогательных модулей для корректной работы.
ESTABLISHED	Состояние ESTABLISHED говорит о том, что это не первый пакет в соединении. Схема установки состояния ESTABLISHED достаточно проста для понимания. Единственное требование, предъявляемое к соединению, заключается в том, что для перехода в состояние ESTABLISHED необходимо, чтобы узел сети передал пакет и получил на него ответ от другого сетевого узла. После получения ответа состояние соединения NEW или RELATED будет изменено на ESTABLISHED.
INVALID	Признак INVALID говорит о том, что пакет не может быть идентифицирован и поэтому не может иметь определенного статуса. Это может происходить по разным причинам, например при нехватке памяти или при получении ICMP-сообщения об ошибке, которое не соответствует какому-либо известному соединению. Наверное, наилучшим вариантом было бы применение действия DROP к таким пакетам.



## 17.4 Действия и переходы iptables

Действия и переходы сообщают правилу, что необходимо выполнить, если пакет соответствует заданному критерию. Чаще всего употребляются действия ACCEPT и DROP. Однако, давайте кратко рассмотрим понятие переходов.

Описание переходов в правилах выглядит точно так же как и описание действий, т.е. ставится ключ `-j` и указывается название цепочки правил, на которую выполняется переход. На переходы накладывается ряд ограничений, первое - цепочка, на которую выполняется переход, должна находиться в той же таблице, что и цепочка, из которой этот переход выполняется, второе - цепочка, являющаяся целью перехода должна быть создана до того как на нее будут выполняться переходы. Например, создадим цепочку `tcp_packets` в таблице `filter` с помощью команды `iptables -N tcp_packets`. Теперь мы можем выполнять переходы на эту цепочку подобно `iptables -A INPUT -p tcp -j tcp_packets`. Т.е. встретив пакет протокола `tcp`, `iptables` произведет переход на цепочку `tcp_packets` и продолжит движение пакета по этой цепочке. Если пакет достиг конца цепочки, то он будет возвращен в вызывающую цепочку (в нашем случае это цепочка `INPUT`) и движение пакета продолжится с правила, следующего за правилом, вызвавшем переход.

Действие - это предопределенная команда, описывающая действие, которое необходимо выполнить, если пакет совпал с заданным критерием. Например, можно применить действие `DROP` или `ACCEPT` к пакету, в зависимости от наших нужд. Существует и ряд других действий, которые описываются ниже в этой секции. В результате выполнения одних действий, пакет прекращает свое прохождение по цепочке, например `DROP` и `ACCEPT`, в результате других, после выполнения некоторых операций, продолжает проверку, например, `LOG`, в результате работы третьих даже видоизменяется, например `DNAT` и `SNAT`, `TTL` и `TOS`, но так же продолжает продвижение по цепочке.

### Действие ACCEPT

Данная операция не имеет дополнительных ключей. Если над пакетом выполняется действие `ACCEPT`, то пакет прекращает движение по цепочке (и всем вызвавшим цепочкам, если текущая цепочка была вложенной) и считается ПРИНЯТЫМ (то есть пропускается), тем не менее, пакет продолжит движение по цепочкам в других таблицах и может быть отвергнут там. Действие задается с помощью ключа `-j ACCEPT`.



## Действие DROP

Данное действие просто "сбрасывает" пакет и *iptables* "забывает" о его существовании. "Сброшенные" пакеты прекращают свое движение полностью, т.е. они не передаются в другие таблицы, как это происходит в случае с действием ACCEPT. Следует помнить, что данное действие может иметь негативные последствия, поскольку может оставлять незакрытые "мертвые" сокеты как на стороне сервера, так и на стороне клиента, наилучшим способом защиты будет использование действия REJECT особенно при защите от сканирования портов.

## Действие RETURN

Действие RETURN прекращает движение пакета по текущей цепочке правил и производит возврат в вызывающую цепочку, если текущая цепочка была вложенной, или, если текущая цепочка лежит на самом верхнем уровне (например, INPUT), то к пакету будет применена политика по умолчанию. Обычно, в качестве политики по умолчанию назначают действия ACCEPT или DROP .

Для примера, допустим, что пакет идет по цепочке INPUT и встречает правило, которое производит переход во вложенную цепочку: --jump EXAMPLE\_CHAIN. Далее, в цепочке EXAMPLE\_CHAIN пакет встречает правило, которое выполняет действие --jump RETURN. Тогда произойдет возврат пакета в цепочку INPUT. Другой пример, пусть пакет встречает правило, которое выполняет действие --jump RETURN в цепочке INPUT. Тогда к пакету будет применена политика по-умолчанию цепочки INPUT.

## Действие LOG

LOG - действие, которое служит для логирования отдельных пакетов и событий. В журнал могут заноситься заголовки IP пакетов и другая интересующая вас информация. Информация из журнала может быть прочитана с помощью *dmesg* или *syslogd* либо с помощью других программ. Превосходное средство для отладки ваших правил. Неплохо было бы на период отладки правил вместо действия DROP использовать действие LOG, чтобы до конца убедиться, что ваш брандмауэр работает безупречно. Обратите ваше внимание так же на действие ULOG, которое наверняка заинтересует вас своими возможностями.

## Действие MARK

- █ Киев    █ Днепропетровск    █ Львов    █ Ровно    █ Мариуполь    █ Полтава
- █ Одесса    █ Донецк    █ Николаев    █ Запорожье    █ Луганск    █ Харьков





Используется для установки меток для определенных пакетов. Это действие может выполняться только в пределах таблицы *tangle*. Установка меток обычно используется для нужд маршрутизации пакетов по различным маршрутам, для ограничения трафика и т.п. Не забывайте, что "метка" пакета существует только в период времени, пока пакет не покинул брандмауэр, т.е. метка не передается по сети. Если необходимо как-то пометить пакеты, чтобы использовать маркировку на другой машине, то можете попробовать манипулировать битами поля TOS.

## Действие REJECT

REJECT используется, как правило, в тех же самых ситуациях, что и DROP, но в отличие от DROP, команда REJECT выдает сообщение об ошибке на хост, передавший пакет. Действие REJECT на сегодняшний день "работает" только в цепочках INPUT, FORWARD и OUTPUT (и во вложенных в них цепочках). Пока существует только единственный ключ, управляющий поведением команды REJECT.

Пример: *iptables -A FORWARD -p TCP --dport 22 -j REJECT --reject-with tcp-reset*

Указывает, какое сообщение необходимо передать в ответ, если пакет совпал с заданным критерием. При применении действия REJECT к пакету, сначала на узел отправителя будет отослан указанный ответ, а затем пакет будет "сброшен". Допускается использовать следующие типы ответов: *icmp-net-unreachable*, *icmp-host-unreachable*, *icmp-port-unreachable*, *icmp-proto-unreachable*, *icmp-net-prohibited* и *icmp-host-prohibited*. По-умолчанию передается сообщение *port-unreachable*. Все вышеуказанные типы ответов являются ICMP error messages. Еще один тип ответа - *tcp-reset*, который используется только для протокола TCP. Если указано значение *tcp-reset*, то действие REJECT передаст в ответ пакет TCP RST, пакеты TCP RST используются для закрытия TCP соединений. (Список типов ICMP ответов и их алиасов вы сможете получить, введя команду *iptables -j REJECT -h*).

## Действие SNAT

SNAT используется для преобразования сетевых адресов (Source Network Address Translation), т.е. изменение исходящего IP адреса в IP заголовке пакета. Например, это действие можно использовать для предоставления выхода в Интернет другим компьютерам из локальной сети, имея лишь один уникальный IP адрес. Для этого необходимо включить пересылку пакетов (forwarding) в ядре и затем создать правила,



которые будут транслировать исходящие IP адреса нашей локальной сети в реальный внешний адрес. В результате, внешний мир ничего не будет знать о нашей локальной сети, он будет считать, что запросы пришли с нашего брандмауэра.

SNAT допускается выполнять только в таблице *nat*, в цепочке *POSTROUTING*. Другими словами, только здесь допускается преобразование исходящих адресов. Если первый пакет в соединении подвергся преобразованию исходящего адреса, то все последующие пакеты, из этого же соединения, будут преобразованы автоматически и не пойдут через эту цепочку правил.

Пример: *iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 194.236.50.155-194.236.50.160:1024-32000*

Ключ *--to-source* используется для указания адреса, присваиваемого пакету. Все просто: вы указываете IP адрес, который будет подставлен в заголовок пакета в качестве исходящего. Если вы собираетесь перераспределять нагрузку между несколькими брандмауэрами, то можно указать диапазон адресов, где начальный и конечный адреса диапазона разделяются дефисом, например: 194.236.50.155-194.236.50.160. Тогда, конкретный IP адрес будет выбираться из диапазона случайным образом для каждого нового потока. Дополнительно можно указать диапазон портов, которые будут использоваться только для нужд SNAT. Все исходящие порты будут после этого преобразовываться в заданный диапазон. *iptables* старается, по возможности, избегать преобразования портов, однако не всегда это возможно. Если диапазон портов не задан, то исходные порты ниже 512 преобразовываются в диапазоне 0-511, порты в диапазоне 512-1023 преобразовываются в диапазоне 512-1023, и, наконец, порты из диапазона 1024-65535 преобразовываются в диапазоне 1024-65535. Что касается портов назначения, то они не подвергаются преобразованию.

## Действие DNAT

DNAT (Destination Network Address Translation) используется для преобразования адреса места назначения в IP заголовке пакета. Если пакет подпадает под критерий правила, выполняющего DNAT, то этот пакет, и все последующие пакеты из этого же потока, будут подвергнуты преобразованию адреса назначения и переданы на требуемое устройство, хост или сеть. Данное действие может, к примеру, успешно использоваться для предоставления доступа к web-серверу, находящемуся в локальной сети, и не имеющему реального IP адреса. Для этого вы строите правило, которое перехватывает пакеты, идущие на HTTP порт брандмауэра и, выполняя DNAT,



передаете их на локальный адрес web-сервера. Для этого действия так же можно указать диапазон адресов, тогда выбор адреса назначения для каждого нового потока будет производиться случайным образом.

Действие DNAT может выполняться только в цепочках PREROUTING и OUTPUT таблицы nat, и во вложенных подцепочках.

Пример: `iptables -t nat -A PREROUTING -p tcp -d 15.45.23.67 --dport 80 -j DNAT --to-destination 192.168.1.1-192.168.1.10`

Ключ `--to-destination` указывает, какой IP адрес должен быть подставлен в качестве адреса места назначения. В выше приведенном примере во всех пакетах, пришедших на адрес 15.45.23.67, адрес назначения будет изменен на один из диапазона от 192.168.1.1 до 192.168.1.10. Как уже указывалось выше, все пакеты из одного потока будут направляться на один и тот же адрес, а для каждого нового потока будет выбираться один из адресов в указанном диапазоне случайным образом. Можно также определить единственный IP адрес. Можно дополнительно указать порт или диапазон портов, на который (которые) будет перенаправлен трафик. Для этого после `ip` адреса через двоеточие укажите порт, например `--to-destination 192.168.1.1:80`, а указание диапазона портов выглядит так: `--to-destination 192.168.1.1:80-100`. Как вы можете видеть, синтаксис действий DNAT и SNAT во многом схож. Не забывайте, что указание портов допускается только при работе с протоколом TCP или UDP, при наличии опции `--protocol` в критерии.

## Действие MASQUERADE

Маскировка (MASQUERADE) применяется в тех же целях, что и SNAT, но в отличие от последней, MASQUERADE дает более сильную нагрузку на систему. Происходит это потому, что каждый раз, когда требуется выполнение этого действия - производится запрос IP адреса для указанного в действии сетевого интерфейса, в то время как для SNAT IP адрес указывается непосредственно. Однако, благодаря такому отличию, MASQUERADE может работать в случаях с динамическим IP адресом, т.е. когда вы подключаетесь к Интернет, скажем через PPP, SLIP или DHCP.

## Действие TTL

Действие TTL используется для изменения содержимого поля Time To Live в IP заголовке. Один из вариантов применения этого действия - это устанавливать значение

- |          |                  |            |             |             |           |
|----------|------------------|------------|-------------|-------------|-----------|
| ■ Киев   | ■ Днепропетровск | ■ Львов    | ■ Ровно     | ■ Мариуполь | ■ Полтава |
| ■ Одесса | ■ Донецк         | ■ Николаев | ■ Запорожье | ■ Луганск   | ■ Харьков |



поля Time To Live ВО ВСЕХ исходящих пакетах в одно и то же значение. Для чего это?! Есть некоторые провайдеры, которые очень не любят, когда одним подключением пользуется несколько компьютеров, если мы начинаем устанавливать на все пакеты одно и то же значение TTL, то тем самым мы лишаем провайдера одного из критериев определения того, что подключение к Интернету разделяется между несколькими компьютерами. Для примера можно привести число TTL = 64, которое является стандартным для ядра Linux.

Действие TTL можно указывать только в таблице *mangle* и нигде больше.

Пример: `iptables -t mangle -A PREROUTING -o eth0 -j TTL --ttl-set 64`

Устанавливает поле TTL в заданное значение. Оптимальным считается значение около 64. Это не слишком много, но и не слишком мало. Не задавайте слишком большое значение, это может иметь неприятные последствия для вашей сети. Представьте себе, что пакет "зацикливается" между двумя неправильно сконфигурированными роутерами, тогда, при больших значениях TTL, есть риск "потерять" значительную долю пропускной способности канала.

## 17.5 Сохранение и восстановление наборов правил

Утилита *iptables-save* предназначена для сохранения текущего набора правил в файл, который затем может быть использован утилитой *iptables-restore*. Эта команда очень проста в использовании и имеет всего два аргумента.

`iptables-save [-c] [-t table]`

Первый аргумент *-c* (допустимо использовать более длинный вариант *--counters*) заставляет *iptables-save* сохранить значения счетчиков байт и пакетов. Это делает возможным рестарт брандмауэра без потери счетчиков, которые могут использоваться для подсчета статистики. По-умолчанию, при запуске без ключа *-c*, сохранение счетчиков не производится.

С помощью ключа *-t* (более длинный вариант *--table*) можно указать имя таблицы для сохранения. Если ключ *-t* не задан, то сохраняются все таблицы. Ниже приведен пример работы команды *iptables-save* в случае, когда набор не содержит ни одного правила.



Утилита *iptables-restore* используется для восстановления (загрузки) набора правил, который ранее был сохранен утилитой *iptables-save*. Список с правилами утилита получает со стандартного ввода и не может загружать его из файла напрямую. Команда имеет следующий синтаксис:

```
iptables-restore [-c] [-n]
```

Ключ *-c* (более длинный вариант *--counters*) заставляет восстанавливать значения счетчиков.

Указание ключа *-n* (более длинный вариант *--noflush*) сообщает *iptables-restore* о том, что правила должны быть добавлены к имеющимся. По-умолчанию утилита *iptables-restore* (без ключа *-n*) очистит содержимое таблиц и цепочек перед загрузкой нового набора правил.

Для загрузки набора правил утилитой *iptables-restore* из файла можно предложить несколько вариантов, но наиболее употребимый:

```
cat /etc/iptables-save | iptables-restore -c
```

В результате выполнения этой команды содержимое файла */etc/iptables-save* будет прочитано утилитой *cat* и перенаправлено на стандартный ввод утилиты *iptables-restore*.

## 17.6 Задание для самопроверки

1. Запретите любые сетевые подключения к вашему компьютеру, за исключением тех, которые были инициированы вашим компьютером.
2. Настройте сервер доступа к сети Интернет в вашей организации, используя технологию NAT. Запретите пользователям вашей сети обращаться на web-узел [www.microsoft.com](http://www.microsoft.com)
3. Выполните задания таким образом, чтобы созданные вами правила автоматически применялись при старте системы.



## 18. Web-сервер Apache

Apache уже долгое время остается самым распространенным web-сервером в сети Интернет. Дистрибутив сервера доступен под различные операционные системы, однако именно Linux считается самой популярной платформой для его использования. Не зря аббревиатура LAMP (Linux, Apache, MySQL, PHP) стала уже именем нарицательным для обозначения web-серверов.

Конфигурирование сервера сводится к редактированию ряда файлов. Основная конфигурация, как правило, хранится в файле *httpd.conf*. В данном файле значительное количество настроек и в некоторых дистрибутивах, для удобства конфигурирования, эти настройки разнесены в разные файлы. Например, так сделано в дистрибутивах SLES.

### 18.1 Виртуальные хосты

Термин «виртуальные хосты» подразумевает такую настройку web-сервера, при которой Apache обслуживает несколько URL, размещенных на одном и том же сервере. Например, несколько доменов, под именами [www.example.com](http://www.example.com) и [www.example.net](http://www.example.net) могут быть запущены под управлением одного web-сервера на одном физическом компьютере. Использование виртуальных хостов позволяет снижать затраты по администрированию web-сервера, так как необходимо управлять только одним сервером. Кроме того, снижаются затраты по приобретению аппаратного обеспечения, так как каждый отдельный домен не требует выделенного физического компьютера.

Рекомендуется всегда использовать конфигурацию с виртуальными хостами, даже если на web-сервере размещается лишь один домен. При такой настройке, конфигурационные параметры специфичные для домена находятся в одном файле. Кроме того, появляется возможность быстрой замены одного набора конфигурационных параметров домена на другой, путем простой замены соответствующего файла. По вышеозначенным причинам следует каждый новый виртуальный хост описывать в отдельном конфигурационном файле.

Рекомендуется настроить виртуальный хост, использующийся по умолчанию. Он будет отображаться для клиентов в ситуациях, когда к серверу обращаются по доменному имени, которое не совпадает с каким-либо виртуальным хостом, обслуживающимся сервером. Виртуальный хост по умолчанию – это хост, чья конфигурация загружается первой (файлы конфигураций сортируются по алфавиту).



## 18.2 /etc/apache2

В данном каталоге находятся конфигурационные файлы web-сервера apache2. Как можно догадаться по названию, это вторая, наиболее актуальная версия данного программного продукта. Конфигурационные файлы в данном каталоге могут быть организованы следующим образом:

Файлы	Назначение
<i>charset.conv</i>	Определяет, какой набор символов используется для различных языков.
<i>conf.d/*.conf</i>	Конфигурационные файлы, добавляемые различными модулями. Эти файлы могут добавляться в конфигурационные файлы различных виртуальных хостов, и каждый виртуальный хост может иметь свой уникальный набор модулей.
<i>default-server.conf</i>	Глобальная конфигурация всех виртуальных хостов. При необходимости изменения указанных здесь настроек для конкретного виртуального хоста следует перезаписывать значения соответствующих параметров в файле его конфигурации.
<i>errors.conf</i>	Как Apache отвечает на различные ошибки. Настройки глобальные для всех хостов. Для настройки параметров отдельного виртуального хоста – необходимо перезаписывать значения соответствующих параметров в файле его конфигурации.
<i>httpd.conf</i>	Основной конфигурационный файл Apache. В зависимости от дистрибутива, может содержать как все настройки сервера, так лишь ссылки на конфигурационные файлы, подгружаемые в данный файл с помощью директивы <i>include</i> .
<i>listen.conf</i>	Задает IP адрес и номер TCP порта, на котором должен быть запущен Apache.
<i>magic</i>	Данные, позволяющие Apache автоматически определять MIME тип файлов.
<i>mime.types</i>	MIME типы известные для всей системы (мягкая ссылка на файл <i>/etc/mime.types</i> ).
<i>mod_*.conf</i>	Конфигурационные файлы модулей, которые проинсталлированы по умолчанию
<i>server-tuning.conf</i>	Конфигурационные директивы, контролирующие производительность Apache.
<i>ssl-global.conf</i> и <i>ssl.*</i>	Настройки SSL и информация о сертификатах.
<i>sysconfig.d/*.conf</i>	Файлы, автоматически генерируемые на основании файла <i>/etc/sysconfig/apache2</i> .
<i>uid.conf</i>	Определяет UID и GID от имени которых должен запускаться Apache.
<i>vhosts.d/*.conf</i>	Каталог с конфигурационными файлами виртуальных хостов.



## 18.3 Виртуальные хосты на основе имени

При настройке виртуальных хостов на основе имени, более одного web-сайта обслуживается на одном IP адресе. Apache сравнивает поле *host* в http-заголовке посылаемом клиентом со значением директивы *ServerName* в описании виртуальных хостов. Если совпадение найдено – клиенту будет подключен к соответствующему хосту. Если нет – подключение будет осуществляться к хосту по умолчанию (первый виртуальный хост в списке хостов).

Директивой *NameVirtualHost* задается на каком IP адресе и, опционально, на каком порте Apache ожидает подключения клиентов к виртуальным хостам. Директива *NameVirtualHost* задается в конфигурационном файле */etc/apache2/listen.conf*. Пример:

```
# NameVirtualHost IP-address[:Port]
NameVirtualHost 192.168.1.100:80
NameVirtualHost 192.168.1.100
NameVirtualHost *:80
NameVirtualHost *
NameVirtualHost [2002:c0a8:364::]:80
```

Для настройки виртуального хоста можно воспользоваться шаблоном, который хранится по адресу */etc/apache2/vhosts.d/vhost.template*. Просто скопируйте данный файл в этот же каталог под новым именем и отредактируйте необходимые значения в полученном файле. Например, таким может быть содержимое конфигурационного файла */etc/apache2/vhosts.d/example.com.conf*

```
<VirtualHost 192.168.1.100>

    ServerName www.example.com
    DocumentRoot /srv/www/www.example.com/htdocs
    ServerAdmin webmaster@example.com
    ErrorLog /var/Log/apache2/www.example.com_Log
    CustomLog /var/Log/apache2/www.example.com-access_Log common

    <Directory "/srv/www/www.example.com/htdocs">
        Order allow,deny
        Allow from all
    </Directory>

</VirtualHost>
```

Разберем представленные опции:

- Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава
- Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков



Директива	Назначение
<code>VirtualHost</code> <code>&lt;/VirtualHost&gt;</code>	Внутри данного тега необходимо указывать значения, предварительно указанные в директиве <code>NameVirtualHost</code> .
<code>ServerName</code>	FQDN хоста.
<code>DocumentRoot</code>	Путь к каталогу, в котором хранятся файлы данного хоста. По соображениям безопасности доступ к файловой системе по умолчанию запрещен, поэтому необходимо предоставлять доступ к содержимому каталога с помощью директивы <code>Directory</code> .
<code>ServerAdmin</code>	E-mail администратора. Этот адрес, в частности, будет отображаться на странице, которую Apache отображает для клиентов при ошибке.
<code>ErrorLog</code>	Файл, в котором будут логироваться ошибки, связанные с этим виртуальным хостом. Хотя нет строгой необходимости в том, чтобы создавать отдельные файлы ошибок для отдельных виртуальных хостов, это рекомендуемая практика, упрощающая работу администратора. <code>/var/log/apache2</code> – каталог, в котором, по умолчанию, хранятся все журнальные файлы сервера.
<code>CustomLog</code>	Файл, в котором будет логироваться информация о доступе клиентов к серверу.
<code>&lt;Directory&gt;</code> <code>&lt;/Directory&gt;</code>	Внутри данного тега указывается каталог, к которому необходимо задать права доступа.
<code>Order</code>	Директива ордер может принимать одно из двух значений: <code>allow,deny</code> или <code>deny,allow</code> .
<code>Order allow,deny</code>	Если клиент не подходит под правила <code>Allow</code> (разрешения доступа) или подходит под правила <code>Deny</code> (запрета доступа), то клиенту будет отказано в доступе.
<code>Order deny,allow</code>	Если клиент не подходит под <code>Deny</code> правила или подходит под <code>Allow</code> правила, ему будет предоставлен доступ.
<code>Allow from all</code>	Разрешить доступ с любого узла. Вместо «all» можно указывать имена доменов ( <code>allow from example.com</code> ), IP адреса узлов ( <code>allow from 10.0.0.1</code> ), адреса сетей ( <code>allow from 192.168</code> )
<code>Deny from all</code>	Запретить доступ с любого узла. Синтаксис аналогичен предыдущему пункту.

После того, как были созданы необходимые файлы конфигураций виртуальных хостов необходимо перезапустить web-сервер:

```
/etc/init.d/apache2 restart
```

Если DNS сервер корректно настроен для разрешения имен виртуальных хостов и функционирует, то клиент получат доступ к ресурсам данных виртуальных хостов.



## 18.4 Модули

Программное обеспечение Apache построено на модульной основе – вся функциональность, за исключением некоторых основных функций, выполняется модулями. Дошло уже даже до того, что http протокол обрабатывается модулем *http\_core*.

Модули Apache могут быть встроены в двоичный файл Apache или могут подгружаться по мере необходимости во время работы сервера. Все модули делятся на четыре основных категории:

Директива	Назначение
<i>Base Modules</i>	Основные модули, интегрированные в двоичный файл Apache по умолчанию. Это, как минимум, модули <i>mod_so</i> (необходим, для подгрузки других модулей) и <i>http_core</i> .
<i>Extension Modules</i>	Модули, которые поставляются в дистрибутиве Apache, но не встроены в двоичный файл Apache, и подгружаются по мере необходимости.
<i>External Modules</i>	Модули, не включенные в официальный релиз Apache.
<i>Multiprocessing Modules</i>	Отвечают за прием и обработку клиентских запросов.

Для активации модуля используется следующая команда:

*a2enmod имя\_модуля*

Для деактивации модуля используется следующая команда:

*a2dismod имя\_модуля*

Для получения списка активных модулей используется следующая команда:

*a2enmod -l*

Назначение каждого модуля и особенности его конфигурирования детально рассмотрены в документации к web-серверу Apache. Познакомиться с ней можно по адресу <http://httpd.apache.org/docs/2.2/mod>.



## 18.5 Настройка SSL

Важные данные, например, информация о кредитной карте или учетные данные электронной почты, должны передаваться между клиентом и web-сервером по защищенным соединениям, с использованием аутентификации и шифрования. Модуль *mod\_ssl* предоставляет необходимую конфиденциальность данных с помощью SSL (Secure Sockets Layer) и TLS (Transport Layer Security) протоколов для взаимодействия клиента и сервера по протоколу HTTP.

Прежде чем клиент посыпает какой-либо запрос на сервер, сервер передает клиенту свой SSL сертификат, с помощью которого клиент имеет возможность установить подлинность сервера, к которому он обращается. Кроме того, сертификат сервера выступает в роли открытого ключа асимметричного шифрования, с помощью которого шифруются данные, передаваемые от клиента к серверу.

Модуль *mod\_ssl* является интерфейсом между web-сервером Apache и SSL библиотекой. В современных Linux системах в качестве SSL библиотеки, как правило, используется OpenSSL.

### 18.5.1 Создание SSL сертификата

Существует три типа сертификатов, которые можно создать. Первый вариант – “dummy” сертификат, который используется только для тестовых целей. Второй вариант – “self-signed” сертификат – сертификат, который мы самостоятельно заверяем и он будет доверенным только для пользователей, которые доверяют нам. Например, это могут быть пользователи нашей организации. Третий вариант – сертификат, подписанный независимым публично известным центром, так называемым Certificate Authority (CA).

#### 18.5.1.1 Создание “dummy” сертификата

Для создания “dummy” сертификата необходимо воспользоваться скриптом */usr/bin/gensslcert*. Он создает или перезаписывает следующие файлы:

*/etc/apache2/ssl.crt/ca.crt*  
*/etc/apache2/ssl.crt/server.crt*



```
/etc/apache2/ssl.key/server.key  
/etc/apache2/ssl.csr/server.csr  
/root/.mkcert.cfg
```

Копия файла *ca.crt* доступна для скачивания клиентами, для чего помещается по следующему адресу: */srv/www/htdocs/CA.crt*. Отметим, что “dummy” сертификат никогда не следует использовать в производственной среде.

### 18.5.1.2 Создание “self-signed” сертификата

Для создания “self-signed” сертификата используется интерактивный процесс, состоящий из девяти шагов. Для запуска данного процесса необходимо перейти в каталог */usr/share/doc/packages/apache2* и выполнить следующую команду:

```
./mkcert.sh make --no-print-directory /usr/bin/openssl /usr/sbin/custom
```

Данный скрипт предоставляет ряд вопросов, на которые необходимо дать ответы. В частности:

***STEP 0: Decide the signature algorithm used for certificates***

Необходимо выбрать “R” (опция по умолчанию), потому что некоторые версии браузеров могут иметь проблемы с использованием алгоритма DSA.

***STEP 1: Generating RSA private key for CA (1024 bit)***

В процессе выполнения данного пункта не требуется вмешательства пользователя.

***STEP 2: Generating X.509 certificate signing request for CA***

На этом этапе необходимо ответить на ряд вопросов, таких как название страны, название организации и т.д. Необходимо вводить достоверные данные, так как они будут отображены в сертификате. Если ответ на какой-то вопрос вы желаете оставить незаполненным, используйте в качестве ответа символ «.». Так как на данном этапе мы создаем сертификат, который будем подписывать самостоятельно, при заполнении опции “Common Name of the CA” необходимо использовать какое-либо выдуманное имя, например “My Company CA”. Это имя должно отличаться от имени вашего узла, не используйте для данного имени FQDN вашего сервера.

***STEP 3: Generating X.509 certificate for CA signed by itself***

Необходимо выбрать сертификат версии 3 (по умолчанию).



## ***STEP 4: Generating RSA private key for SERVER (1024 bit)***

В процессе выполнения данного пункта не требуется вмешательства пользователя.

## ***STEP 5: Generating X.509 certificate signing request for SERVER***

Вопросы на этом этапе практически идентичны вопросам с Шага №2. Важным этапом данного пункта является то, что в секции “Common Name” необходимо указать FQDN вашего сервера, например [www.example.com](http://www.example.com). В противном случае, браузер клиентов будет выдавать сообщение о том, что сертификат не совпадает с именем web-сервера.

## ***STEP 6: Generating X.509 certificate signed by own CA***

Необходимо выбрать сертификат версии 3 (по умолчанию).

## ***STEP 7: Encrypting RSA private key of CA with a pass phrase for security***

По соображениям безопасности строго рекомендуется шифровать частный ключ СА с помощью пароля, таким образом, рекомендуется ответить “Y” и ввести пароль.

## ***STEP 8: Encrypting RSA private key of SERVER with a pass phrase for security***

Шифрование ключа сервера требует ввода пароля при каждом запуске web-сервера. Это усложняет автоматический запуск web-сервера при загрузке или перезапуске сервера. Поэтому, обычно на данном этапе выбирают “N” в качестве ответа на заданный вопрос. При этом необходимо помнить, что частный ключ сервера остается незащищенным и необходимо строго контролировать список пользователей, авторизованных доступа к ключу. Если было решено, все же, шифровать ключ, то необходимо увеличить опцию APACHE\_TIMEOUT в файле /etc/sysconfig/apache2, так как в противном случае, у администратора не будет достаточное количество времени для ввода пароль, прежде чем сервер остановит свой запуск по причине ошибки.

По итогам выполнения скрипта будут сгенерированы 2 сертификата и 2 ключи, по 1 паре сертификат-ключ для СА и web-сервера. Они будут размещены в соответствующие подкаталоги каталога /etc/apache2.

Последний этап – копирование СА сертификата из каталога /etc/apache2/ssl.crt/ в каталог, доступный для пользователей вашей организации. Каждый клиент должен в своем браузере добавить данный сертификат в список доверенных издателей. Если этого не будет сделано, браузеры клиентов будут сообщать о том, что сертификат для web-сервера был выдан неизвестным источником. Срок действия сертификата – 1 год.



### 18.5.1.3 Официально подписанный сертификат

Существуют различные официальные СА, которые могут подписать ваш сертификат. Это компании Thawte ([www.thawte.com](http://www.thawte.com)), Verisign ([www.verisign.com](http://www.verisign.com)), Comodo ([www.comodo.com](http://www.comodo.com)). Информация об этих СА уже встроена в современные браузеры, так, что они автоматически признают сертификаты подписанные данными компаниями.

Для запроса подписанного сертификата, отсылается не сам сертификат, а так называемый CSR (Certificate Signing Request). Для его создания необходимо запустить скрипт:

```
/usr/share/ssl/misc/CA.sh -newreq
```

После того, как вы ответите на ряд вопросов в интерактивном режиме, создастся файл *newreq.pem*, который и будет вашим CSR. При ответах на вопросы указывайте достоверную информацию, так как все, что вводится в качестве ответов, будет отображаться в информации о сертификате. Для пустых полей, традиционно, необходимо использовать символ «..».

### 18.5.2 Настройка Apache на использование SSL

Порт по умолчанию, использующийся web-сервером для приема SSL запросов имеет номер 443. Таким образом, не существует никакого конфликта между использованием HTTP и HTTPS протоколов одновременно одним сервером.

По умолчанию SSL модуль активирован, но если это необходимо сделать вручную, используется команда

```
a2enmod SSL
```

Для того чтобы сервер запускался с включенным SSL, необходимо его запускать с флагом "SSL". Для этого необходимо выполнить следующую команду:

```
a2enflag SSL
```

В SLES необходимо проследить за тем, чтобы в файле */etc/sysconfig/apache2* была указана следующая строка:



APACHE\_SERVER\_FLAGS="SSL"

Для настройки виртуальных хостов с поддержкой SSL можно использовать шаблон, который хранится по адресу `/etc/apache2/vhosts.d/vhost-ssl.template`. Хотя, как правило, на одном IP адресе следует запускать лишь один web-узел с поддержкой SSL. В противном случае, клиенты будут получать предупреждения о том, что имя узла не соответствует информации, указанной в сертификате.

Если вам необходим виртуальный хост, на который следует сразу обращаться по защищенному соединению, с использованием https протокола, то вы необходимо создать виртуальный хост на 80 порту вашего сервера, который будет перенаправлять клиентов на 443 порт. Например, для узла `www.oleg.local` необходимая конфигурация виртуальных хостов будет выглядеть следующим образом:

```
mail:/etc/apache2/vhosts.d # cat _www80.oleg.local.conf
<VirtualHost *:80>
    ServerName www.oleg.local
    Redirect permanent / https://www.oleg.local/
</VirtualHost>
mail:/etc/apache2/vhosts.d # cat _www.443oleg.local.conf
<VirtualHost *:443>
    SSLEngine On
    SSLCertificateFile    /etc/apache2/ssl.crt/server.crt
    SSLCertificateKeyFile /etc/apache2/ssl.key/server.key

    ServerName www.oleg.local
    DocumentRoot /srv/www/htdocs/www.oleg.local/wordpress/
    <Directory "/srv/www/htdocs/www.oleg.local/wordpress">
        Order allow,deny
        Allow from all
    </Directory>

</VirtualHost>
```

Конфигурацию файла `/etc/apache2/listen.conf` следует обновить одной строкой:





```
mail:/etc/apache2/vhosts.d # tail /etc/apache2/listen.conf
#
NameVirtualHost *:80
NameVirtualHost *:443
```

Итак, виртуальным хостом по умолчанию будет <https://www.oleg.local>. Если у вас настроены другие http хосты, они будут работать без изменений.

## 18.6 Задание для самопроверки.

1. Настройте виртуальный хост [www.example.local](http://www.example.local). Настройте виртуальный хост [wp.example.local](http://wp.example.local), на который пользователи обращаются по https протоколу. Настройте хостом по умолчанию для web-сервера узел [https://wp.example.local](http://wp.example.local).
2. Для отображения контента при обращении на [https://wp.example.local](http://wp.example.local) установите CMS Wordpress.





## 19. Прокси-сервер Squid

Squid – широко используемый кэширующий прокси-сервер для Linux платформ. Информация, которую клиенты получают с web или ftp серверов, сохраняется на сервере, который находится гораздо ближе к клиентам. Объекты, которые запрашиваются клиентами, сохраняются в дисковом кэше прокси-сервера. Если несколько клиентов запрашивают один и тот же объект, то для них этот объект может быть загружен из дискового кэша. Это позволяет доставлять данные для клиентов с гораздо большей скоростью и приводит к уменьшению внешнего сетевого трафика вашей организации.

Помимо функции кэширования, Squid предоставляет ряд дополнительных функций: аутентификация пользователей, ограничение доступа к определенным ресурсам, ограничение скорости получения информации, получение статистики об использовании Internet ресурсов. Squid работает лишь с некоторыми протоколами, основные из них - это HTTP и FTP. В связке с брандмауэром прокси-сервер создает безопасное взаимодействие с ресурсами в сети Internet. Брандмауэр запрещает любые клиентские обращения к Internet ресурсам, за исключением прокси-сервера Squid. Таким образом, пользователи внутренней сети могут только с использованием Squid обращаться к Internet ресурсам, что позволяет установить полный контроль web доступа клиентов вашей организации.

Не все объекты, к которым обращаются пользователи, являются статическими. Динамически изменяемые объекты не кешируются, так как они каждый раз изменяются при новой попытке доступа к ним. Статические ресурсы также могут меняться со временем и, как следствие, всегда будет актуальным следующий вопрос: как долго следует хранить в кэше объект? Для ответа на этот вопрос каждый объект в кэше снабжается специальным заголовком, в котором хранится информация о том, когда он был последний раз изменен или какой срок актуальности данного объекта. Объекты в кэше, как правило, вытесняют друг друга согласно алгоритму LRU (Last Recently Used) – прокси-сервер удаляет объект из кэша, если к нему не было долгое время обращений, и заменяет его более актуальным объектом.

Запуск и остановка прокси-сервера Squid осуществляется традиционно:

```
/etc/init.d/squid start  
/etc/init.d/squid stop
```

■ Киев   ■ Днепропетровск   ■ Львов   ■ Ровно   ■ Мариуполь   ■ Полтава  
■ Одесса   ■ Донецк   ■ Николаев   ■ Запорожье   ■ Луганск   ■ Харьков



## 19.1 Аппаратные требования

Требования к аппаратному обеспечению сервера, на котором используется прокси-сервер Squid – отдельная тема обсуждения. Прокси-сервер Squid активно использует большинство ресурсов сервера – жесткие диски, оперативную память, процессор, сетевые интерфейсы. Недостаточная производительность какого-либо компонента системы может привести к неудовлетворительной работе прокси-сервера.

### 19.1.1 Требования к дисковой подсистеме

Основное требование к жестким дискам для работы прокси-сервера Squid состоит в том, что дисковая подсистема должна отличаться хорошими значениями параметра *random seek time* (время поиска случайного блока жесткого диска). Данный параметр вычисляется в миллисекундах. Блоки данных, которые записывает или считывает Squid достаточно мало, и скорость обращения к случайному блоку жесткого диска в таком случае гораздо важнее пропускной способности интерфейса жесткого диска. В таком случае, для прокси-сервера лучше подходят жесткие диски, с большим количеством оборотов в минуту (RPM), так как скорость обращения к случайному блоку у них выше. Для увеличения скорости работы дисковой подсистемы, можно использовать несколько дисков, объединенных в RAID-0. Отказоустойчивость дисковой подсистемы в данном случае не так важна, так как данные, хранящиеся в кэше, не являются той информацией, о безопасности которой необходимо беспокоиться.

### 19.1.2 Требования к размеру кэша

При выборе размера кэша происходит традиционный конфликт интересов. Маленький кэш лучше тем, что в нем быстрее происходит поиск нужной прокси-серверу информации. Большой кэш лучше тем, что в него помещается больше информации, и, соответственно, чем больше информации помещено в кэш, тем меньше внешнего трафика генерируется клиентами.

Самый простой способ вычисления необходимого размера кэша – вычисление максимальной скорости передачи Internet соединения. Если скорость подключения к сети – 1 Мб/с, то скорость передачи данных – 125 КБ/с. Если весь такой трафик сохраняется в кэше, то за один час работы прокси-сервера дисковый кэш будет заполнен на 450 МБ. За один рабочий день (восемь рабочих часов) такой трафик приведет к заполнению дискового кэша на 3,6 ГБ. Учитывая, что среднестатистический



трафик, как правило, не соответствует максимально возможной скорости подключения, то можно предположить, что в нашем примере дисковый кэш за один рабочий день заполнится на 2ГБ. Вот по каким причинам значение 2ГБ рекомендуется разработчиками Squid как размер, позволяющий эффективно кэшировать данные, к которым обращаются клиенты прокси-сервера в течение одного рабочего дня.

### 19.1.3 Требования к оперативной памяти

Количество необходимой для прокси-сервера оперативной памяти напрямую зависит от количества объектов, хранящихся в кэше. Объекты, к которым клиенты обращаются наиболее часто, помещаются в оперативную память, так как скорость ее работы несравненно больше скорости работы дисковой подсистемы. Кроме того, в оперативной памяти хранятся не только кэшированные объекты, но и другие данные, например таблица обслуживаемых прокси-сервером IP адресов и списки контроля доступа. В случае если оперативной памяти для прокси-сервера будет недостаточно, скорость его работы уменьшится в несколько раз, так как работы с данными из swap происходит во много раз медленнее.

### 19.1.4 Требования к центральному процессору

Прокси-сервер Squid не относится к разряду программ, активно использующих процессор. В данном случае использование многопроцессорных систем не увеличит производительность сервера. Таким образом, для увеличения производительности необходимо больше внимания уделить объемам оперативной памяти и скорости дисковой подсистемы, нежели скорости работы процессора.

## 19.2 /etc/squid/squid.conf

Все настройки прокси-сервера Squid хранятся в конфигурационном файле */etc/squid/squid.conf*. Данный конфигурационный файл снабжен подробнейшими комментариями и примерами использования того или иного параметра. Рассмотрим некоторые параметры файла */etc/squid/squid.conf* и примеры их конфигурации.

#### *http\_port 3128*

Задает порт, на котором прокси-сервер Squid ожидает обращения клиентов. По умолчанию используется порт 3128, также обычной практикой является запуск прокси-сервера на порте под номер 8080.



### ***cache\_mem 8 MB***

Данный параметр задает объем оперативной памяти, используемый прокси-сервером Squid для хранения наиболее популярных объектов. Значение по умолчанию – 8 МБ.

### ***maximum\_object\_size 4096 KB***

Данный параметр задает максимально допустимый размер объекта, помещаемого в дисковый кэш.

### ***cache\_dir ufs /var/cache/squid 100 16 256***

Задает каталог, в котором будет храниться дисковый кэш прокси-сервера Squid. В данном каталоге будет сформирована специальная файловая система, тип которой – ufs. Под каталог будет выделено 100 МБ дискового пространства. В каталоге будет создано 16 подкаталогов, в каждом из которых будет создано 256 подкаталогов. При выборе размера каталога, следует руководствоваться количеством свободного пространства на соответствующем разделе жесткого диска. Рекомендуется использовать от 50% до 80% свободного пространства. Для хранения кэша одновременно на нескольких дисках, необходимо использовать несколько опций *cache\_dir*.

### ***cache\_access\_log /var/Log/squid/access.log***

### ***cache\_log /var/Log/squid/cache.log***

### ***cache\_store\_log /var/Log/squid/store.log***

Эти три параметра задают адрес файлов, куда будут логироваться все данные о работе прокси-сервера Squid. Как правило, значения, принятые здесь по умолчанию, не требуют каких-либо изменений. Однако если прокси-сервер испытывает серьезные нагрузки, можно задать расположение файлов кэша и журнальных файлов на разные жесткие диски.

### ***cache\_peer hostname type proxy-port icp-port***

В данной опции можно указать адрес родительского прокси-сервера, например адрес прокси-сервера провайдера. В опции *hostname* – указывается IP адрес или FQDN имя родительского прокси-сервера, в опции *type* – значение *parent*. В опции *proxy-port* – порт, на котором запущен родительский прокси-сервер. Если *icp* порт неизвестен, в опции *icp-port* следует использовать значение 0 или 7.

### ***cache\_mgr webmaster***

Адрес электронной почты администратора сервера. На этот адрес Squid отсылает письмо, если прокси-сервер непредвиденно завершил свою работу.



#### ***client\_netmask 255.255.255.255***

С помощью данной опции можно маскировать IP адреса клиентов в журнальных файлах прокси-сервера Squid. Например, если задать маску 255.255.255.0, то во всех IP адресах клиентов последний байт адреса будет заменен на нулевое значение. Так можно обеспечить конфиденциальность пользователей.

#### ***ftp\_user Squid@***

При аутентификации на различных FTP серверах под анонимной учетной записью, часто требуется отправить адрес электронной почты в качестве пароля. В данной опции можно задать адрес, который и будет использоваться в качестве пароля. Параметр по умолчанию – *Squid@* может вызывать проблемы при попытках аутентификации, так как данный адрес не пройдет проверку на «правильный» почтовый адрес.

#### ***append\_domain domain***

В случае если клиент отправил запрос на сервер, в имени которого не указан домен, то к запросу будет автоматически добавлен суффикс, указанный в данной опции. Как правило, задается доменный суффикс вашей организации. Например, example.com. И, если пользователь в строке браузера ввел адрес <http://www>, то такой адрес будет преобразован в <http://www.example.com>.

#### ***forwarded\_for on***

По умолчанию прокси-сервер Squid добавляет в HTTP запрос клиентов строку такого вида: *X-Forwarded-For: 192.168.0.1*. Если переключить значение рассматриваемой опции в *off*, то данная информация не будет включаться прокси-сервером в перенаправляемый HTTP запрос.

#### ***negative\_ttl 5 minutes***

Задает количество времени, на срок которого в кэше хранятся негативные ответы, такие как “Connection Refused” или “404 not found”.

Перезапуск прокси-сервера Squid, как правило, занимает довольно длительное время ввиду того, что Squid должен закрыть все соединения клиентов и синхронизировать данные, хранящиеся в оперативной памяти и в дисковом кэше. Поэтому, для того, чтобы новые настройки файла */etc/squid/squid.conf* вступили в силу, вместо перезапуска сервера необходимо перезагружать его конфигурацию:

```
/etc/init.d/squid reload
```



## 19.3 Контроль доступа к прокси-серверу

Прокси-сервер Squid предоставляет удобный и эффективный механизм контроля доступа с помощью списков контроля доступа (Access Control Lists, ACL). С помощью ACL задается список объектов, например пользователей или узлов. По умолчанию уже определены такие ACL как *all* (все клиенты прокси-сервера) и *localhost*. Недостаточно просто создать ACL для того, чтобы они «работали». Созданные или уже существующие ACL необходимо указывать в директивах, управляющих доступом к прокси-серверу, например в правилах, указанных с помощью опции *http\_access*. Рассмотрим примеры ACL (опция *-i* в описании ACL отключает регистрозависимость регулярных выражений):

```
acl mysurfers srcdomain .my-domain.com
```

# ACL по имени “mysurfers” задает  
# клиентов, обращающихся к прокси-  
# серверу из домена mydomain.com

```
acl teachers src 192.168.1.0/255.255.255.0
```

# ACL по имени “teachers” задает  
# клиентов, обращающихся к прокси-  
# серверу из сети 192.168.1.0/24

```
acl Lunch time MTWHF 12:00-15:00
```

# ACL по имени “Lunch” задает  
# диапазон времени с 12:00 до 15:00  
# в будние дни

```
acl students src 192.168.7.0-192.168.9.0/255.255.255.0
```

# “students” – клиенты сервера,  
# обращающиеся из сетей  
# 192.168.7.0-192.168.9.0/24

```
acl mp3files urlpath_regex -i \.mp3$
```

# “mp3files” – все \*.mp3 файлы

Рассмотрим примеры использования ACL в директиве *http\_access*:

```
http_access allow Localhost
```

# Разрешить полный Internet доступ  
# при обращении с Localhost

```
http_access allow teachers
```

# Разрешить доступ к ресурсам сети  
# Internet клиентам, подходящим  
# под ACL “teachers”

```
http_access allow students Lunch
```

# Ограничить доступ к ресурсам сети  
# Internet клиентам, подходящим  
# под ACL “students” только будними  
# днями с 12:00 до 15:00

```
http_access deny all
```

# Всем остальным клиентам  
# (не определенным выше)  
# доступ запрещен



## 19.4 Аутентификация пользователей

Для более тонкой настройки прокси-сервера, необходим инструмент, с помощью которого клиенты могут проходить аутентификацию с использованием имени пользователя и пароля. Непосредственно Squid не имеет встроенных механизмов для этого, однако можно воспользоваться внешними программами.

Если пользователи для аутентификации должны использовать учетные записи, созданные на сервере, то для проверки имени пользователя и пароля следует применять программу *pam\_auth*. Если список пользователей прокси-сервера должен храниться отдельно от локальных учетных записей сервера, то можно воспользоваться утилитой *ncsa\_auth*. Также есть возможность аутентификации пользователей с помощью учетных записей Active Directory под управлением Windows серверов. Такая процедура выполняется с помощью утилиты *squid\_ldap\_auth*.

### 19.4.1 pam\_auth и ncsa\_auth

Для указания программы, с помощью которой прокси-сервер проводит аутентификацию клиентов, используется опция *auth\_param*. Например:

```
auth_param basic program /usr/sbin/pam_auth
```

После того, как была задана программа аутентификации, следует настроить доступ к серверу таким образом, чтобы только пользователи прошедшие аутентификацию смогли получить доступ к Internet ресурсам:

```
acl password proxy_auth REQUIRED
```

```
http_access allow password
http_access deny all
```

Опция *REQUIRED* может быть заменена на список имен пользователей, которым разрешен доступ к прокси-серверу, или на адрес файла, содержащего такой список.

Для использования программы *ncsa\_auth* следует создать файл, в котором будут храниться пользователи, и, затем, добавить в него пользователей с паролями. Для этого используется утилита *htpasswd2*. Например:





```
htpasswd2 -c /etc/squid/squid.users student
htpasswd2 /etc/squid/squid.users teacher
```

Первой командой был создан файл и в него был добавлен пользователь *student*. Второй командой произошло простое добавление пользователя, по этой причине не был указан ключ “-c”. В опции *auth\_param* необходимо указать адрес программы *ncsa\_auth* и путь к созданному файлу с пользователями:

```
auth_param basic program /usr/sbin/ncsa_auth /etc/squid/squid.users
```

#### 19.4.2 Active Directory аутентификация

Для настройки аутентификации пользователей с использованием их учетных записей в Active Directory, необходимо определить учетную запись, от имени которой Squid будет проводить поиск пользователей в Active Directory каталоге. Для проверки того, успешно ли данный Squid от имени данной учетной записи проводит поиск, необходимо выполнить следующую команду:

```
/usr/sbin/squid_ldap_auth -R -D squidreader@domain.com -W /etc/squid/adpw.txt \
-b "dc=domain,dc=com" -f "sAMAccountName=%s" -h dc.domain.com
```

В данной команде: *domain.com* – имя Active Directory домена вашей организации, *squidreader* – учетная запись, от имени которой Squid проводит поиск пользователей в Active Directory каталоге, *dc.domain.com* – контроллер домена, */etc/squid/adpw.txt* – файл, в котором хранится пароль пользователя *squidreader*.

Признаком успешного выполнения команды является то, что в новой строке нам предоставляется возможность ввести имя Active Directory пользователя и его пароль (через пробел). В результате должен быть получен ответ “OK”. После этого можно переходить к следующему пункту действий – добавлению следующей строки в */etc/squid/squid.conf*:

```
auth_param basic program /usr/sbin/squid_ldap_auth -R -D \
squidreader@domain.com -W /etc/squid/adpw.txt -b "dc=domain,dc=com" -f \
"sAMAccountName=%s" -h dc.domain.com
```

В итоге все пользователи, у которых есть учетные записи в Active Directory домене вашей организации, получают доступ к Internet ресурсам посредством прокси-сервера.





## 19.4.2.1 Предоставление доступа группе пользователей

Если на предприятии есть необходимость в предоставлении доступа к Internet ресурсам только определенной Active Directory группе пользователей, то необходимо выполнить несколько шагов в настройке такой конфигурации.

**ШАГ 1.** Создание группы пользователей. Например, группу под названием *InetUsers*, находящуюся в подразделении (OU) *Groups* нашего домена.

**ШАГ 2.** Необходимо убедиться в том, что Squid в состоянии проверить членство пользователя в группе:

```
/usr/sbin/squid_ldap_group -R -b "dc=domain,dc=com" -f "(&(sAMAccountName=%v) \ 
(memberOf=cn=%a,ou=Groups,dc=domain,dc=com))" -D squidreader@domain.com -W \
/etc/squid/adpw.txt -h dc.domain.com
```

После ввода команды в новой строке необходимо ввести имена пользователя и группы, разделенные пробелом, например:

*Boss InetUsers*  
*OK*

Если в результате был получен ответ “OK”, можно переходить к следующему шагу настройки.

**ШАГ 3.** В */etc/squid/squid.conf* добавляем следующую строку:

```
external_acl_type ad_users %LOGIN /usr/sbin/squid_ldap_group -R -b \
"dc=domain,dc=com" -f "(&(sAMAccountName=%v) \ 
(memberOf=cn=%a,ou=Groups,dc=domain,dc=com))" -D squidreader@domain.com -W \
/etc/squid/adpw.txt -h dc.domain.com
```

**ШАГ 4.** В */etc/squid/squid.conf* добавляем ACL:

```
acl inetusers external ad_users InetUsers
```

В данном примере *inetusers* – имя ACL, а *InetUsers* – имя Active Directory группы.

**ШАГ 5.** Настраиваем необходимые настройки *http\_access*. Например:





```
http_access allow inetusers
http_access deny all
```

В итоге, только пользователи, являющиеся членами Active Directory группы *InetUsers*, смогут обращаться к Internet ресурсам посредством прокси-сервера Squid.

## 19.5 Дележ внешнего канала между пользователями

Внешний канал, по умолчанию, делится равномерно между всеми клиентами прокси-сервера. Однако в организации может быть необходимость таких настроек, при которых одним пользователям предоставляется большая скорость работы с Internet ресурсами, а другим – меньшая. Такая технология в терминологии прокси-сервера Squid называется *delay pools*.

Рассмотрим пример самой простой настройки дележа внешнего канала, при которой два разных клиента получают в свое распоряжение два разных канала с разными ограничениями скорости в них:

```
acl teacher src 10.0.6.139/32      # ACL “teacher” задает клиента, с IP адресом 10.0.6.139
acl student src 10.0.6.138/32       # ACL “student” задает клиента, с IP адресом 10.0.6.138

delay_pools 2                         # Мы настраиваем 2 канала, которые для клиентов будут
                                      # предоставлять разные допустимые скорости работы

delay_class 1 1                      # создаем канал №1, класс которого – первый.

delay_class 2 1                      # создаем канал №2, класс которого – первый.

delay_access 1 allow teacher          # доступ к каналу №1 получает только клиент “teacher”
delay_access 1 deny all               # остальным пользователям доступ к нему запрещен

delay_access 2 allow student          # доступ к каналу №2 получает только клиент “student”
delay_access 2 deny all               # остальным пользователям доступ к нему запрещен

delay_parameters 1 64000/64000        # канал №1 имеет ограничение скорости 64КБ/с.
                                      # ограничение вступает в силу сразу после того, как
                                      # клиент скачал 64КБ информации

delay_parameters 2 640/640            # канал №2 имеет ограничение скорости 640Б/с.
                                      # ограничение вступает в силу сразу после того, как
                                      # клиент скачал 640Б информации
```



В предыдущем примере использовался только первый класс канала, в котором задается ограничение трафика для всего ACL. Можно использовать также второй и третий класс канала. Второй класс позволяет указывать отдельно ограничения трафика для подсети и для каждого узла в данной подсети. Третий класс позволяет ограничивать трафик сети класса В, подсетей класса С и каждого отдельного узла подсетей. В зависимости от масштабов организации, могут пригодиться тот или иной класс канала.

Для того чтобы в «свойствах» канала использовались не IP адреса узлов, а имена пользователей, прошедших аутентификацию, следует использовать следующий набор ACL и настроек каналов:

```
acl password proxy_auth REQUIRED
acl student proxy_auth username student
acl teacher proxy_auth username teacher

delay_pools 2

delay_class 1 1
delay_class 2 1

delay_access 1 allow teacher
delay_access 1 deny all

delay_access 2 allow student
delay_access 2 deny all

delay_parameters 1 64000/64000
delay_parameters 2 640/640
```

## 19.6 Настройка прозрачного прокси-сервера

Для работы пользователей с прокси-сервером, необходимо настраивать браузеры клиентов. Это не всегда можно сделать централизовано, и по этой причине бывает удобным настройка прозрачного прокси-сервера, при котором не требуется какой-либо конфигурации клиентских браузеров.

На сервере включается NAT и перенаправление клиентских запросов с 80-го TCP порта на TCP порт под номером 3128 (или другой порт, на котором запущен прокси-сервер Squid). Такую настройку можно выполнить с помощью *iptables*:





```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 3128
```

Для настройки прокси-сервера Squid необходимо в */etc/squid/squid.conf* в опции *http\_port* добавить аргумент *transparent*:

```
http_port 3128 transparent
```

Существует еще ряд сценариев, при которых в организации понадобится использование прозрачного прокси-сервера. Например, по соображениям безопасности все клиенты должны использовать прокси-сервер, или для шейпинга трафика (delay pools в терминах прокси-сервера Squid).

Итак, принцип работы прозрачного прокси-сервера очень прост. Прокси-сервер перехватывает все запросы клиентов и самостоятельно генерирует ответы. Пользователь не догадывается о том, что между ним и web-сервером находится прокси-сервер, именно по этой причине такая конфигурация называется прозрачной.



## 20. Файловый сервер на базе Samba

В конце 1990-х в локальных сетях все чаще стали устанавливать компьютеры под управлением Linux. Увлеченные перспективой затрачивать для решения сложных задач относительно небольшие ресурсы, администраторы использовали Linux даже тогда, когда это было совершенно не оправдано. Часто такими компьютерами заменяли дорогие и ненадежные серверы Windows, в результате пришлось решать задачу взаимодействие клиентов, работающих под управлением Windows, и Linux-серверов. Возникла потребность в специальном инструменте, позволяющем поддерживать новую конфигурацию сети.

Таким инструментом стал продукт сервер Samba, обеспечивающий поддержку протокола SMB (Server Message Block - блок сообщений сервера), который в настоящее время известен также под названием CIFS (Common Internet Filesystem - общая межсетевая файловая система). SMB/CIFS - это протокол, позволяющий организовывать совместное использование файлов и принтеров, и работающий на базе NetBIOS (набор протоколов, широко используемый в сетях, содержащих компьютеры под управлением Windows). Другими словами, Samba позволяет компьютеру под управлением Linux выполнять функции файлового сервера и сервера печати в сетях, содержащих компьютеры Windows. В настоящее время Samba очень хорошо справляется с этой задачей, кроме того, данный продукт постоянно дорабатывается и дополняется новыми средствами.

В простейших случаях работу сервера Samba можно организовать, не внося существенных изменений в содержимое конфигурационных файлов. Несмотря на это, Samba считается чрезвычайно сложным продуктом, на работу которого оказывают влияние многочисленные параметры. Подробное описание формата конфигурационного файла *smb.conf* приведено в справочной системе; для того чтобы получить необходимую информацию, нужно выполнить команду *man smb.conf*.

Несмотря на то, что сервер Samba может выполнять различные функции, прежде всего, он представляет собой инструмент для предоставления в общий доступ файлов и принтеров. На клиентской машине можно смонтировать часть файловой системы удаленного компьютера и обращаться к ней как к локальной файловой системе. С файлами, находящимися в файловой системе удаленной машины, могут работать различные приложения на клиентском компьютере, например, некоторый файл можно загрузить в текстовый редактор, внести в него изменения и сохранить на сервере.





Подобные операции часто выполняются в сетевой среде офиса. Средства монтирования каталогов удаленной машины позволяют хранить на сервере файлы с данными и коды приложений. Используя принтеры, предоставленные в общий доступ, пользователи могут выводить данные на устройства печати, подключенные к серверам. Организация пула принтеров позволяет сэкономить ресурсы.

Поскольку NetBIOS и SMB/CIFS наследуют некоторые характеристики систем DOS и Windows, очевидно, что Samba целесообразнее всего применять в сетях, использующих компьютеры под управлением именно этих операционных систем. Ряд свойств Samba упрощает обмен данными с DOS и Windows. Например, всем известно, что в именах файлов DOS и Windows не учитывается регистр символов; имена FILE.TXT, file.txt File.txt представляют один и тот же файл. В Linux, напротив, имена файлов зависят от регистра символов. Обеспечивая совместную работу Windows и Linux, Samba позволяет обращаться к файлам без учета регистра. Кроме того, SMB/CIFS поддерживает такие особенности файловой системы DOS и Windows, как атрибуты скрытых файлов и файлов архивов. В файловой системе Linux скрытые файлы и файлы архивов отсутствуют, но Samba позволяет устанавливать и использовать соответствующие атрибуты.

Подобные требования предъявляются при обмене файлами с другими операционными системами, например системой OS/2, поэтому серверы Samba могут быть использованы и для работы с ними. Samba применяется даже в сетях, не использующих компьютеры под управлением DOS, Windows, OS/2 и других систем, в которых основным протоколом разделения файлов является SMB/CIFS. UNIX, Macintosh, BeOS и другие системы также поддерживают SMB/CIFS. Если такая поддержка не реализована в самой системе, она обеспечивается продуктами независимых производителей. Linux позволяет работать и с другими протоколами разделения файлов (в частности, Linux поддерживает средства NFS, которые будут подробно рассмотрены позже), однако в некоторых случаях использование Samba предпочтительнее. Модель защиты SMB/CIFS, в которой для аутентификации применяются пользовательские имена и пароли, отличается от модели NFS, где компьютеры идентифицируются по IP-адресам, а за безопасность системы отвечают средства защиты клиента.

Сервер Samba включает в себя три основных службы: *smbd* для SMB/CIFS служб, *nmbd* для службы имен и *winbind* для аутентификации. Для запуска сервера необходимо выполнить команды:

```
/etc/init.d/smb start && /etc/init.d/nmb start && /etc/init.d/winbind start
```



## 20.1 /etc/samba/smb.conf

Данный файл является основным в конфигурировании сервера Samba. Данный файл разделен на две логические части. Первая часть – секция *[global]*, в которой хранятся глобальные настройки сервера. Вторая часть – это все секции *[share]*, в которых содержится информация об объектах, предоставляемых в общий доступ клиентам. Это могут быть как каталоги, так и принтеры.

### 20.1.1 Секция [global]

Параметры, которые будут описаны ниже, должны быть заполнены администратором для корректной работы сервера Samba в сети предприятия.

#### *workgroup = TUX-NET*

Эта строка задает имя рабочей группы, в которую входит сервер Samba. Вместо “TUX-NET” следует указать имя рабочей группы сетевого окружения вашего предприятия. Сам сервер будет доступен в рабочей группе по DNS имени. Если DNS сервер не используется на предприятии, следует указать имя сервера следующей опции.

#### *netbiosname = MYNAME*

Эта строка задает NetBIOS имя сервера. Клиенты смогут обращаться к серверу по имени или по IP-адресу. В производственной среде предпочтительнее использовать DNS имя для сервера Samba.

#### *os Level = 20*

Когда в сети нет выделенного WINS сервера, хранящего список всех узлов рабочей группы, один из таких узлов должен быть выбран в качестве LMB (local master browser). Если в сети предприятия есть Windows серверы, предпочтительнее настроить сервер Samba так, чтобы он не был выбран случайным образом в качестве LMB. Для этого следует понизить «приоритет» сервера Samba в текущем параметре. Например, изменив значение по умолчанию (20) до значения «2». Если, по каким-либо причинам, вы хотите чтобы именно сервер Samba хранил список узлов рабочей группы, необходимо увеличить это значение, например до «65».

#### *wins server*

Если в Windows сети используется WINS сервер, его IP адрес следует указать в аргументах данного параметра. Это позволит интегрировать сервер Samba в сеть.



### wins support

Для того чтобы сервер Samba исполнял роль WINS сервера, необходимо в аргументе данного параметра указать значение "Yes". При этом необходимо убедить в том, что на других серверах Samba вашей сети данный параметр не активирован. Кроме того, следует помнить, что параметры *wins server* *wins support* не могут быть активированы одновременно в файле *smb.conf*.

### 20.1.2 Секции [share]

Рассмотрим примеры предоставления в общий доступ домашних каталогов:

<i>[homes]</i>	; имя ресурса, предоставляемого в общий доступ: "homes". ; Однако это имя будет автоматически заменяться именем ; пользователя после его успешной аутентификации. С помощью ; данного ресурса пользователь получает возможность ; обратиться в свой домашний каталог.
<i>comment = Home Directories</i>	; комментарий, в котором указывается назначение ресурса
<i>valid users = %S</i>	; %S – имя каталога, предоставляемого в общий доступ. ; В нашем примере пользователи обращаются в свои ; домашние папки, чье имя совпадает с именем пользователя
<i>browsable = No</i>	; ресурс не будет отображаться в списке доступных ресурсов
<i>read only = No</i>	; по умолчанию все ресурсы предоставляются в общий доступ ; в режиме «только для чтения». Для включения возможности ; записи можно использовать указанный параметр, который ; является синонимом параметра <i>writeable = Yes</i>
<i>create mask = 0640</i>	; права доступа, которые будут назначены создаваемым файлам. ; Значение "0640" означает права: Владелец – чтение и запись. ; Основная группа владельца - только чтение.
<i>directory mask = 0750</i>	; аналогично предыдущему, но применяется к новым каталогам.

Рассмотрим пример предоставления в общий доступ CD привода:

<i>[cdrom]</i>	; имя ресурса
<i>comment = Linux CD-ROM</i>	; комментарий, в котором указывается назначение ресурса
<i>path=/media/cdrom</i>	; путь к каталогу, куда смонтирован компакт-диск
<i>guest ok = Yes</i>	; доступ к приводу разрешен любому пользователю сети





Таким образом, при обращении к общему ресурсу по имени “cdrom” пользователи будут обращаться к каталогу, который на локальной файловой системе находится по адресу */media/cdrom*. Опция “path=” необходима для всех каталогов, предоставляемых в общий доступ, за исключением ресурса “homes”, относительно которого в сервере Samba действует специальное соглашение.

### 20.1.3 Сценарии postexec и preeexec

Samba поддерживает параметры *preeexec* и *postexec*, которые позволяют выполнять некоторые команды при регистрации пользователя и завершении его работы с разделяемым объектом. В качестве значения параметра *preeexec* задаются команды, которые должны быть выполнены при регистрации пользователя, соответственно команда, указанная как значение *postexec*, выполняется при завершении работы пользователя с объектом. Например, если вы хотите, чтобы при обращении к разделяемому объекту сервер Samba передавал почтовое сообщение по адресу *admin@itstep.org* вы должны включить в определение этого объекта следующее выражение:

```
preeexec = mail -s "share being used" admin@itstep.org
```

Если пользователь зарегистрируется для работы с объектом, сервер Samba пошлет от его имени сообщение по адресу *admin@itstep.org*. В поле Subject сообщения будет включена строка "Share being used", а по адресу отправителя получатель сможет выяснить, кто из пользователей работал с объектом.

Аналогично действует параметр *postexec*, но команда, заданная в качестве его значения, выполняется после окончания работы с объектом. Зная особенности работы Windows-клиентов с разделяемыми объектами SMB/CIFS, можно сделать вывод, что команда не будет выполнена сразу же после того, как пользователь закроет окно, открытое с помощью Network Neighborhood или My Network Places, но через некоторое время обязательно произойдет.

Разновидностями параметров *preeexec* и *postexec* являются параметры “*root preeexec*” и “*root postexec*”. Отличаются они лишь тем, что команды, заданные в качестве значений *root preeexec* и *root postexec*, выполняются от имени пользователя root. Таким образом, можно задавать команды, для выполнения которых требуются специальные привилегии. Используя эти параметры, следует соблюдать осторожность.



При выполнении сценариев сервер Samba может обрабатывать переменные, пере-численные ниже. Эти переменные позволяют настроить сценарии *preexecs* и *postexecs* для работы с конкретными пользователями, клиентами, операционными системами, установленными на клиентских компьютерах, и т.д. Рассмотрим некоторые переменные, которые доступны в сервере Samba:

Переменная	Назначение
%a	Операционная система на клиентском компьютере
%d	Идентификатор процесса сервера
%g	Основная группа, к которой относится пользователь, указанный в переменной %u
%G	Основная группа, к которой относится пользователь, указанный в переменной %Г
%h	FQDN имя сервера
%H	Рабочий каталог пользователя, указанного в переменной %u
%I	IP адрес клиента
%L	NetBIOS имя сервера
%m	NetBIOS имя клиента
%M	Конфигурационные директивы, контролирующие производительность Apache
%P	Имя каталога, связанного с ресурсом, предоставленным в общий доступ
%S	Имя разделяемого объекта
%T	Текущая дата и текущее время
%u	Эффективное имя пользователя Linux
%U	Имя пользователя, зарегистрированного в Linux (может не совпадать с %u)

Параметры *preexecs* и *postexecs* в основном предназначены для того, чтобы задавать команды, подготавливающие разделяемые объекты к использованию. Так, например, если есть опасность, что пользователь, работающий в системе Windows, по ошибке удалит конфигурационный файл Linux, сценарий *preexecs* можно использовать для создания резервной копии этого файла. В целом, параметры *preexecs* и *postexecs* применяют для решения самых разнообразных задач.



## 20.2 Безопасность сервера и ресурсов

Сервер Samba предусматривает множество механизмов увеличения безопасности сервера и ресурсов, предоставляемых в общий доступ. Защита ресурсов собственным паролем, аутентификация пользователей с использованием логина и пароля, списки пользователей, которым разрешен или запрещен доступ к ресурсам – эти и другие механизмы будут рассмотрены в данном разделе.

### 20.2.1 Уровни безопасности сервера

Сервер Samba предоставляет несколько, так называемых, Security Levels:

#### *Share Level Security (security = share)*

Для каждого отдельного ресурса задается собственный пароль. Каждый, кто знает данный пароль, может использовать данный ресурс.

#### *User Level Security (security = user)*

Каждый пользователь должен зарегистрироваться на сервере, используя свой логин и пароль. Сервер предоставляет доступ пользователю к определенном ресурсам, основываясь на том, какой у пользователя логин.

#### *Server Level Security (security = server)*

Для клиентов сервер Samba работает в *user mode*. Для этого необходимо задать параметр "*password server*".

#### *ADS Level Security (security = ADS)*

В этом режиме сервер Samba выступает в роли члена Active Directory домена. Для работы в этом режиме, необходимо, чтобы был установлена и сконфигурирована служба Kerberos. Также необходимо присоединить сервер Samba в домен.

#### *Domain Level Security (security = domain)*

Этот режим работает корректно только в том случае, если сервер Samba был введен в домен под управлением Windows NT.

Выбор безопасности производится на уровне сервера, невозможно определить уровень безопасности для конкретного ресурса. Если в этом все же есть необходимость, следует запустить несколько экземпляров служб Samba на разных IP адресах, которые сконфигурированы в вашей системе.



## 20.2.2 Безопасность ресурсов

Для контроля доступа к ресурсам, предоставляемым в общий доступ, используется следующие параметры:

Параметр	Назначение
<i>read only</i> =	Параметр "Yes" – доступ к ресурсу предоставляется только в режиме чтения Параметр "No" – доступ к ресурсу предоставляется в режиме чтения и записи
<i>writable</i> =	"Yes" – доступ в режиме чтения и записи, "No" – только чтение
<i>valid users</i> =	Список пользователей (через пробел), которым разрешен доступ к ресурсу
<i>invalid users</i> =	Список пользователей (через пробел), которым запрещен доступ к ресурсу
<i>write list</i> =	Список пользователей (через пробел), которым разрешена операция записи
<i>read list</i> =	Список пользователей (через пробел), которым разрешена операция чтения
<i>hosts allow</i> =	Список узлов (через пробел), с которых разрешен доступ к ресурсу. Пример: <i>hosts allow</i> = 192.168.1. ns.itstep.org
<i>hosts deny</i> =	Список узлов (через пробел), с которых запрещен доступ к ресурсу

Обратите внимание на параметры "*write list*" и "*read list*". Эти параметры позволяют переопределять для отдельных пользователей установки, разрешающие чтение и запись или только чтение. Предположим, что вы создали разделяемый объект и поместили программные файлы. Очевидно, что подавляющее большинство пользователей не должны вносить изменения в содержимое этого объекта, поэтому данный разделяемый объект целесообразно определить как предназначенный только для чтения. Однако некоторые пользователи будут заниматься обновлением программ, поэтому им надо предоставить права записи. Этих пользователей можно определить с помощью параметра *write list*. В качестве примера применения описанных выше параметров рассмотрим фрагмент конфигурационного файла:

```
[Program]
; имя ресурса
path = /opt/program
; локальный каталог, на который ссылается ресурс
read only = Yes
; для большинства пользователей - только для чтения
invalid users = guest, student
; но для пользователей guest и student - доступ запрещен
write list = boss, teacher
; а пользователям boss и teacher - разрешена запись
```



## 20.3 Samba как сервер входа

В сетях, где большинство компьютеров работают под управлением Windows, предпочтительнее использовать такой принцип предоставления ресурсов в общий доступ, при котором пользователи регистрируются на сервере с использованием верного имени пользователя и пароля. Как правило, в Windows – сетях используется PDC (Primary Domain Controller), который отвечает за централизованную аутентификацию пользователей в Windows – сетях. Однако такую же задачу может выполнять и Linux система с помощью сервера Samba. Для этого необходимо выполнить несколько шагов.

### ШАГ 1. Редактируем параметры в глобальной секции:

```
[global]
Workgroup = TUX-NET
domain Logons = Yes
domain master = Yes
```

### ШАГ 2. Создаем учетные записи пользователей и компьютеров:

В глобальной секции активируем использование зашифрованных паролей:

```
[global]
encrypt passwords = yes
```

Создаем пользователя и задаем ему пароль:

```
useradd -m teacher
smbpasswd -a teacher
```

Создаем учетную запись компьютеров, как этого требует концепция Windows домена, и задаем ему пароль:

```
useradd hostname\$
smbpasswd -a -m hostname
```

Для более подробных инструкций следует обращаться к документу под названием Samba3-HOWTO, расположенного по адресу */usr/share/doc/packages/samba/Samba3-HOWTO.pdf* (12 глава). Для этого необходимо предварительно установить пакет под названием *samba-doc*.



## 20.4 SWAT

Для администрирования сервера Samba можно использовать Web-интерфейс, который предоставляется средством, под названием SWAT (Samba Web Administration Tool). Консоль управления доступна по адресу <http://localhost:901>. Но для этого должен быть запущен демон *swat*, запуск которого, как правило, контролируется супердемоном *xinetd*.

Для активации *swat* следует отредактировать файл */etc/xinetd.d/swat*, в котором параметру “*disable*” необходимо задать значение “*no*”. После этого необходимо перезапустить службу *xinetd*:

```
/etc/init.d/xinetd restart
```

После перезапуска, запускаем браузер, вводим адрес <http://127.0.0.1:901> и наслаждаемся результатом.

## 20.5 Задания для самопроверки

1. Создайте общий ресурс *incoming*, к которому пользователи из группы *teachers* имеют полный доступ, а остальные пользователи - только право записи.
2. Создайте общий ресурс *public*, в котором каждый пользователь может создавать свои объекты и не может удалять чужие.





## 21. Почтовый сервер на базе Postfix

Postfix – это агент передачи сообщений (*MTA, message transport agent*), который занимается пересылкой по протоколу SMTP сообщений от пользовательского почтового агента (*MUA, mail user agent*), называемого также почтовым клиентом, к удаленному почтовому серверу. МТА также принимает сообщения от удаленных почтовых серверов и пересыпает их другим МТА или доставляет в локальные почтовые ящики. Переслав или доставив сообщение, Postfix заканчивает свою работу. За доставку сообщения конечному пользователю отвечают другие серверы, которые принято называть МАА (*mail access agent*). Например, такие МАА, как серверы POP3 или IMAP, передают сообщения почтовым клиентам – Mutt, Outlook или Apple Mail, с помощью которых пользователь может прочитать их.

На первый взгляд работа МТА кажется совсем простой, но это не так. Особенность агентов передачи сообщений заключается в том, что им приходится пересыпать информацию через границы сетей – они передают данные в другие сети и принимают данные в своей сети. Здравый смысл подсказывает, что каждый, кто использует сеть, должен принимать необходимые меры предосторожности и защищать свои серверы и данные от возможных атак. Широко распространенное мнение о том, что для этого достаточно установить межсетевой экран, контролирующий соединения между локальной и удаленными сетями в обоих направлениях, не более чем миф: межсетевой экран – это не программа, а концепция.

Самая известная часть типичной реализации межсетевого экрана – это приложение, которое контролирует и ограничивает соединения. К сожалению, межсетевые экраны, как правило, не способны анализировать содержимое сообщений, которыми обмениваются хосты; они могут контролировать сами хосты, порты и протоколы транспортного уровня, используемые при передаче данных, но не могут ограничивать соединения, основываясь на их содержимом. Анализ передаваемых данных – намного более сложная задача, требующая применения специализированных программ, способных определить, является ли содержимое вредоносным, и решить, что с ним делать. Для электронной почты эту задачу решают МТА. Кроме того, современные МТА должны быть быстрыми, надежными и безопасными, т. к. они отвечают за передачу электронной почты – наиболее распространенных в Интернете данных.

Безопасность в Postfix основана на его настройках по умолчанию. Если приложение в базовой конфигурации безопасно и достаточно функционально, чтобы не требовать

- |          |                  |            |             |             |           |
|----------|------------------|------------|-------------|-------------|-----------|
| ■ Киев   | ■ Днепропетровск | ■ Львов    | ■ Ровно     | ■ Мариуполь | ■ Полтава |
| ■ Одесса | ■ Донецк         | ■ Николаев | ■ Запорожье | ■ Луганск   | ■ Харьков |





дополнительного вмешательства, то получить на его основе безопасный МТА не составит труда. Простой структурированный синтаксис описания параметров Postfix позволяет с легкостью модифицировать поведение, установленное по умолчанию. Кроме того, Postfix имеет модульную структуру, в которой каждый отдельный модуль выполняется с наименьшим уровнем привилегий, позволяющим ему работать.

Витсес Венема, разработчик Postfix, говорят, что "Postfix – это фактически маршрутизатор, который выбирает пути для сообщений, а не для IP пакетов".

## 21.1 Подготовка хоста и окружения

В Postfix не дублируется функциональность, которая имеется в Linux по умолчанию. В силу этого Postfix рассчитывает, что ваша система правильно настроена – его работоспособность напрямую определяется работоспособностью операционной системы. Рассмотрим контрольный список действий.

### Правильно укажите имя хоста

Для надежного взаимодействия с другими системами почтовый сервер должен иметь полностью определенное имя (FQDN), например mail.example.com. В своем приветствии удаленным почтовым клиентам и серверам Postfix автоматически подставляет это данное вами имя, если только вы вручную не задали ему другое. Полностью определенное доменное имя важно еще и потому, что функции Postfix не ограничиваются приемом почты от клиентов – в клиентском режиме Postfix также передает сообщения другим почтовым серверам. Многие почтовые серверы проверяют заявленное клиентом имя и не принимают сообщения, если имя не является полностью определенным, а некоторые еще и проверяют, разрешается ли указанное имя в DNS.

Операционная система устанавливает имя хоста во время загрузки. Чтобы проверить, есть ли уже у вашей системы полностью определенное имя, используйте команду *hostname*:

```
hostname -f
```

Если эта команда не вернула вам полностью определенное доменное имя, найдите, где оно устанавливается в вашей системе, и исправьте его. Если же такое имя у вашей системы есть, но вы хотели бы использовать в Postfix другое, оставьте настройку



системы как есть. Значение по умолчанию вы замените при помощи параметра *myhostname* в конфигурационном файле Postfix.

Чтобы задать системе hostname отредактируйте файл */etc/HOSTNAME* и впишите туда желаемый вами FQDN вашего сервера.

### Проверьте способность Postfix устанавливать соединения

Убедитесь, что ваша машина может выйти в сеть и что другие хосты этой сети могут общаться с ней. Первая часть не вызывает затруднений: достаточно открыть онлайновую веб-страницу, чтобы убедиться, что ваш компьютер в сети. Входящие соединения проверить немного сложнее. Для этого вам понадобится другой хост в сети, с которого клиенты могли бы устанавливать соединение. Если Postfix предлагает услуги всему Интернету, вам следует проверить возможность соединения с хостом, полностью независимого от вашего сервера.

Убедитесь, что порт 25 TCP на вашем сервере ничем не заблокирован. Если у вас установлен межсетевой экран, убедитесь, что его политика разрешает входящие и исходящие соединения с портом 25. Имейте в виду, что некоторые интернет-провайдеры блокируют на своих маршрутизаторах исходящие соединения с портом 25, если вы явно не попросите снять это ограничение. Провайдер может отказаться отменить данный запрет и предложит вам свои почтовые серверы в качестве ретрансляторов, например с использованием SMTP-аутентификации.

Причина, по которой порт 25 TCP должен быть открыт, заключается в том, что Postfix и другие почтовые серверы прослушивают его в ожидании соединений. Этот порт официально назначен для SMTP агентством IANA (полный список доступен по адресу <http://www.iana.org/assignments/port-numbers>). Организация IANA является главным регистратором нумерации в интернет-протоколах, распределяющим номера портов.

### Убедитесь, что сообщения Postfix записываются в системный журнал

Одно из основных мест, где следует искать диагностические сообщения, – это почтовый журнал. В Postfix используется стандартная для Linux утилита регистрации *syslog* (или *syslog-ng*). Настройка *syslog* была подробно рассмотрена нами в одной из предыдущих глав. Напомним, что сообщения от почтовых служб регистрируются в *syslog* с помощью средства *mail*. Вот как могут быть настроены файлы, в которые попадают сообщения от данного средства:





```

#
# Mail-messages in separate files:
#
destination mailinfo { file("/var/log/mail.info"); };
log { source(src); filter(f_mailinfo); destination(mailinfo); };

destination mailwarn { file("/var/log/mail.warn"); };
log { source(src); filter(f_mailwarn); destination(mailwarn); };

destination mailerr { file("/var/log/mail.err" fsync(yes)); };
log { source(src); filter(f_mailerr); destination(mailerr); };

```

К сожалению, есть несколько типичных проблем, которые могут возникнуть при работе с *syslog*. Если вы не видите в журнале ни одной записи, прежде всего надо проверить, действительно ли запущена служба *syslog* (*syslog-ng*). Например, так:

```
sles:~/Desktop # ps aux | grep syslog
root      1813  0.0  0.0  2628   840 ?          Ss   12:56   0:01 /sbin/syslog-ng
 -a /var/lib/dhcp/dev/log -a /var/lib/named/dev/log
root      5257  0.0  0.0  3496   804 pts/0      S+   13:46   0:00 grep syslog
```

Кроме того, прежде чем давать указание *syslogd* использовать журнальные файлы, убедитесь, что они существуют и доступны для записи. В некоторых реализациях *syslog* не предусмотрено автоматическое создание журнальных файлов и сообщений об ошибках при наличии проблем с ними.

## Настройте разрешение имен для клиента

Прежде чем Postfix, как и любой почтовый сервер, сможет передать сообщение удаленному адресату, он должен определить его местоположение. В Интернете поиск удаленных ресурсов осуществляется с помощью службы доменных имен (DNS). Сервер имен возвращает IP адрес, соответствующий доменному имени, и наоборот, доменное имя, соответствующее указанному IP адресу.

Хорошо работающая служба DNS критически важна для производительности МТА. Чем скорее Postfix сможет получить искомый IP адрес, тем раньше он свяжется с удаленным почтовым сервером и начнет передачу сообщения. Низкая производительность службы имен может стать главным тормозом в больших почтовых системах. В случае перебоев в работе сервера DNS может помочь кэширующий сервер. Устанавливайте кэширующий сервер имен в больших почтовых системах. Имейте в





виду, что применение мер против спама может потребовать увеличения количества запросов почтового сервера к DNS в несколько раз.

Прежде чем пытаться увеличить производительность разрешения имен в своей системе, убедитесь, что оно корректно выполняется в вашей операционной системе, запросив у сервера имен запись MX. Попробуйте выполнить такую команду:

```
sles:~/Desktop # dig google.com MX

; <>> DiG 9.5.0-P2 <>> google.com MX
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25534
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;google.com.           IN      MX

;; ANSWER SECTION:
google.com.          900     IN      MX      400 google.com.s9b2.psmtp.com.
google.com.          900     IN      MX      100 google.com.s9a1.psmtp.com.
google.com.          900     IN      MX      200 google.com.s9a2.psmtp.com.
google.com.          900     IN      MX      300 google.com.s9b1.psmtp.com.

;; Query time: 80 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Jan 12 14:10:46 2011
;; MSG SIZE  rcvd: 162
```

Результат показывает, что используются четыре почтовых сервера, принимающих почту для адресатов, находящихся в домене google.com. На некоторых устаревших платформах утилита *dig* не входит в стандартную поставку. Вы можете найти ее в составе дистрибутива BIND на сайте консорциума ISC (<http://www.isc.org>). Если вы не можете установить *dig*, то, возможно, вам удастся выполнить данный запрос с помощью команд *host* или *nslookup*; хотя последнюю использовать уже не рекомендуется.

Итак, если поиск выполнен успешно, Postfix (теоретически) может корректно выполнять разрешение имен. Если же запрос не выполнен и имена хостов не определяются, вам необходимо срочно разобраться с проблемами DNS.

Одна из распространенных проблем с разрешением имен вызвана попытками обращения к недоступному серверу имен. Проверьте свой файл */etc/resolv.conf*.





Предположим, в нем есть записи такого вида, согласно которым ваш хост обращается к серверу имен на *localhost* (127.0.0.1), а затем, не найдя его там, обращается к хосту 8.8.8.8:

```
nameserver 127.0.0.1
nameserver 8.8.8.8
```

Все хорошо, если на локальной машине запущен кэширующий сервер имен. Но если его там нет, запрос будет ждать окончания таймаута. Если впоследствии выяснится, что разрешение имен в команде *dig* работает, а Postfix тем не менее не может найти хост (например, в журнале присутствует запись “*unknown host*”), то скорее всего Postfix запущен с помощью *chroot* и использует другой файл конфигурации при разрешении имен. Например, если вы указали в команде *chroot* путь /var/spool/postfix, то Postfix будет обращаться к файлу /var/spool/postfix/etc/resolv.conf. В таком случае, с помощью команды *cpan -p /etc/resolv.conf /var/spool/postfix/etc/resolv.conf* убедитесь, что файлы соответствуют друг другу, затем перезапустите Postfix.

## Создайте в DNS записи для почтового сервера

Вы должны настроить свой сервер имен таким образом, чтобы он мог сообщить остальному миру, что доставкой почты в данном домене занимается именно ваш почтовый сервер. Системный администратор, который отвечает за работу сервера имен в вашем домене, должен добавить следующие записи в файл зоны домена:

### A-запись

Ваш почтовый сервер должен иметь полное доменное имя, чтобы клиенты могли его найти. Запись типа A устанавливает соответствие между доменным именем и IP адресом.

### PTR-запись

Имя вашего хоста должно поддаваться обратному разрешению. Почтовые серверы, получившие это имя в сеансе SMTP, должны иметь возможность убедиться, что они действительно работают с вашим сервером.

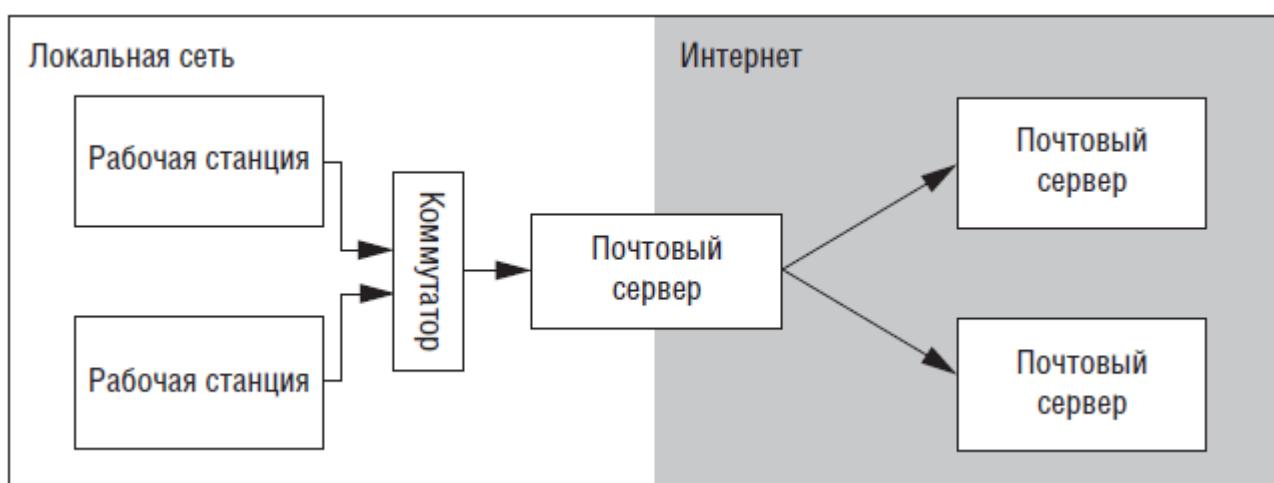
### MX-запись

Записи типа MX сообщают клиентам, что ваш сервер отвечает за доставку почты для домена или определенного хоста.



## 21.2 Настройка почтового сервера одного домена

Настройка Postfix для обслуживания одного домена – дело нескольких минут. Независимо от того, какую конфигурацию вы планируете в дальнейшем, на первом этапе всегда начинайте с конфигурации для единственного домена: так вы сможете убедиться, что Postfix работает в наиболее простом варианте. Так выглядит типичная архитектура сети для минимальной конфигурации:



Почтовый сервер имеет постоянное соединение с Интернетом и статический IP адрес. Прямые (типа A) и обратные (типа PTR) записи DNS для почтового сервера созданы. Мы настроим Postfix на прием почты для одного домена, чтобы он доставлял сообщения с разными адресами в пределах этого домена в разные почтовые ящики. В задачи Postfix будет входить обработка почты только для этого домена. Мы приведем минимальный набор изменений, которые необходимо выполнить в конфигурации после типовой установки. Наш компьютер будет называться *mail.itstep.org*, а домен – *itstep.org*. Выполним несколько шагов для настройки Postfix.

### 21.2.1 Настройка имени хоста в заголовке smtpd

Когда почтовый клиент и сервер «встречаются», они приветствуют друг друга, называя свои DNS имена. Первое, что мы сделаем, – это определим имя, с которым Postfix будет представляться почтовым клиентам. Если имя вашего хоста совпадает с тем, которое вы хотите использовать в приветствии, вам повезло: ничего менять не надо.





Если же в вашей системе установлено имя хоста `www.itstep.org`, а вы хотите, чтобы Postfix, работающий на этой же машине, использовал в своем приветствии имя `mail.itstep.org`, то в этом нет ничего сложного.

Есть два способа назначить другое имя: задать значение параметра `myhostname` или параметра `mydomain`.

### Параметр `myhostname`

Чтобы установить значение параметра `myhostname`, отредактируйте файл `/etc/postfix/main.cf`. Откройте этот файл в любом редакторе и найдите строку `myhostname`. Затем введите полностью определенное доменное имя хоста:

```
myhostname = mail.itstep.org
```

Как только вы задали параметр `myhostname`, Postfix может автоматически получить значение `mydomain`. Postfix просто отбрасывает все до первой точки включительно. А поскольку мы задали параметру `myhostname` значение `mail.itstep.org`, Postfix установит в `mydomain` значение `itstep.org` – что нам и требовалось.

### Параметр `mydomain`

Можно не использовать параметр `myhostname`, а ограничиться установкой `mydomain`. Такой вариант может оказаться очень удобным, если вам надо растиражировать конфигурацию на несколько машин.

```
mydomain = example.com
```

Как только вы задали параметр `mydomain`, Postfix может получить значение `myhostname`, объединив вывод команды `uname -n` на данном хосте со значением `mydomain`. Из этого следует, что, если в файле `main.cf` явно задан только параметр `mydomain` и вы копируете его на другие машины в этом же домене (`itstep.org` в нашем примере), Postfix самостоятельно сформирует правильное имя хоста.

Почему так важно установить правильное значение параметра `myhostname`? Дело в том, что когда Postfix передает сообщения другим почтовым серверам, он выступает в качестве почтового клиента. Представляясь почтовому серверу, он по умолчанию использует параметр `myhostname` в приветствии `HELO`. Некоторые почтовые серверы





настроены так, что отвергают сообщение, если имя в *HELO* не соответствует полностью определенному доменному имени, полученному обратным разрешением. Либо убедитесь, что имя хоста, заданное для Postfix, соответствует имени, получаемому по IP адресу вашего сервера, либо установите значение параметра *smtp\_helo\_name* в соответствии с официальным именем из пространства имен DNS.

## 21.2.2 Настройка домена, в который адресована почта

Для локальных клиентов Postfix будет служить ретранслятором – в том смысле, что он будет принимать почту для тех доменов, в которых он не является конечным или промежуточным сервером. Все, что вам нужно сделать в конфигурации с одним доменом, – это задать значение параметра *mydestination*.

Наша цель состоит в том, чтобы научить Postfix принимать любую почту, адресованную в домен *itstep.org*. В силу того, что мы уже присвоили это значение параметру *mydomain*, можем просто сослаться на него при задании значения *mydestination* в файле *main.cf*:

```
mydestination = $mydomain
```

Если вы хотите тем же способом заставить Postfix принимать почту для хоста, указанного в параметре *myhostname*, то просто добавьте этот параметр в список *mydestination*:

```
mydestination = $mydomain, $myhostname
```

Как видите, значения в списке разделяются запятыми, запятая в конце отсутствует. Продолжая в том же духе, вы можете добавить в список *www.itstep.org* и *ftp.itstep.org*, объединив имена хостов с параметром *\$mydomain*:

```
mydestination =  
$mydomain,  
$myhostname,  
www.$mydomain,  
ftp.$mydomain
```

Этот пример также иллюстрирует другую форму записи. Если в параметре надо указать много значений, то каждое из них можно поместить в отдельной строке, но при этом каждая строка должна начинаться с пробела (иначе Postfix не сможет распознать





значение). Такой формат может быть использован для любого параметра Postfix, способного принимать несколько значений. Вы можете проверить это в окне командного интерпретатора с помощью следующей команды:

```
postconf mydestination
```

### 21.2.3 Настройка домена, добавляемого в исходящие сообщения

Когда локальная программа, такая как *cron*, *at* или работающий в командной строке почтовый клиент, отправляет почту, она обычно не указывает полный адрес отправителя или получателя, ограничиваясь именем пользователя. Это вполне допустимо для локальных адресатов, но при отправке сообщения на другой хост возникает проблема.

Выяснение того, откуда пришло сообщение, занимает довольно много времени, а в случае отсутствия на указанном хосте нужного адресата принимающий почтовый сервер не сможет вернуть сообщение.

У Postfix есть параметр, значение которого добавляется к адресу отправителя или получателя, если он указан не полностью: *myorigin*. Как и раньше, вы можете ссылаться на параметры, ранее определенные в *main.cf*:

```
myorigin = $mydomain
```

Как только этот параметр вступит в силу, Postfix будет добавлять значение из *mydomain* к любому адресу, если он задан не полностью. Например, сообщение от процесса *cron* пользователю *root* получит адрес *root@\$mydomain*, что в нашем случае будет преобразовано в *root@itstep.org*.

Если вы не укажете значение *myorigin*, то по умолчанию будет подставляться значение параметра *myhostname*, что может быть удобно, если у вас есть несколько хостов, для которых сообщения от *root* должны доставляться на один адрес на центральном сервере. В таком случае вы всегда будете знать, от какого хоста получено сообщение; в сообщении *cron*, например, Postfix превратит *root* в *root@\$myhostname*, что в нашем случае будет преобразовано в *root@mail.itstep.org*.



## 21.2.4 Перенаправление сообщений для root в другой почтовый ящик

Postfix доставляет почту непосредственно локальным пользователям, включая и пользователя *root*, но в процессе доставки внешние программы не смогут получить привилегии *root*. Это означает, что вы не можете использовать локальные агенты доставки (*LDA – local delivery agents*), такие как *procmail* или *maildrop*, для доставки почты пользователю *root*, т. к. Postfix не будет запускать их с привилегиями *root*. Вместо этого он запустит их с привилегиями *default\_privs*, по умолчанию соответствующими привилегиям пользователя *nobody*. Такая мера предосторожности предусмотрена для того, чтобы никоим образом не скомпрометировать учетную запись суперпользователя, запуская с его привилегиями уязвимые внешние программы. Но это не значит, что пользователю *root* нельзя доставить адресованные ему сообщения. Решение заключается в создании отдельного пользователя с обычными низкими привилегиями и перенаправлении ему адресованных суперпользователю сообщений.

В нашем примере мы используем имя *admin* для учетной записи, под которой осуществляется администрирование сервера. Чтобы Postfix доставлял почту для *root* пользователю *admin*, просто откройте файл */etc/aliases*, созданный при установке по умолчанию, и впишите следующую строку:

```
root: admin
```

Файл *aliases*, входящий в дистрибутив Postfix, содержит все адреса, которые должны быть на почтовом сервере согласно различным RFC. В самом файле вы найдете указания, где искать более подробную информацию об этих требованиях.

Важное замечание для нашего примера: чтобы не получилось зацикливание, удостоверьтесь в том, что в файле *aliases* нет записи *admin: root*.

Когда вы отредактировали файл */etc/aliases* и добавили туда нужного пользователя, вам надо создать индексированную версию – обычно с именем */etc/aliases.db*, – чтобы ускорить поиск. Для этого нужно либо обработать файл */etc/aliases* программой *postalias*, либо запустить *newaliases* без параметров. Чтобы воспользоваться возможностями Postfix, выполните такую команду:

```
postalias hash:/etc/aliases
```



## 21.2.5 Запуск Postfix

Пришло время выполнить первые тесты. В предыдущем разделе мы добавили и изменили ряд параметров, и если будем двигаться дальше, не удостоверившись, что все работает правильно, то рискуем в дальнейшем столкнуться с трудностями при поиске ошибки.

Прежде чем начать отправлять почту, мы должны запустить Postfix. Все, что для этого нужно, ввести следующую команду:

```
postfix start
```

На что Postfix ответит таким сообщением:

```
postfix/postfix script: starting the Postfix mail system.
```

Если же сообщение будет таким:

```
postfix/postfix script: fatal: the Postfix mail system is already running
```

Значит, Postfix уже работает.

Если Postfix был запущен в то время, когда вы занимались изменением конфигурации, то эти изменения не повлияют на его работу. Можно было бы остановить и снова запустить Postfix, чтобы заставить его прочитать конфигурацию, но для этого есть более элегантный способ. Просто введите такую команду:

```
postfix reload
```

И в ответ получите следующее сообщение:

```
postfix/postfix script: refreshing the Postfix mail system.
```

В этом случае Postfix загрузит только конфигурацию, что займет меньше времени и не прервет обслуживание клиентов.

Аналогичные действия можно выполнять и с помощью сценария, находящегося по адресу `/etc/init.d/postfix`, который принимает аргументы `start`, `stop`, `reload` и т.д.



## 21.2.6 Отправка тестового сообщения

Теперь, когда Postfix запущен, можно выполнить первый тест: доставить почту, адресованную *root*, в его почтовый ящик. Это можно сделать двумя очень простыми способами: отправить сообщение из командной строки или из сеанса *telnet*. Оба способа хороши тем, что позволяют исключить влияние других приложений, таких как сложные почтовые клиенты с графическим интерфейсом, и позволяют сконцентрироваться на Postfix в случае возникновения ошибок.

### Отправка сообщения из Postfix - версии sendmail

Самый простой и надежный способ проверки базовой функциональности – использовать программу *sendmail*; в этом случае используются только компоненты Postfix. Эта утилита командной строки названа *sendmail* с целью сохранения совместимости – многие отправляющие почту приложения в Linux содержат в себе жестко заданную ссылку на программу *sendmail* в виде */usr/sbin/sendmail* или */usr/lib/sendmail*. Туда же Postfix помещает свою программу *sendmail*, чтобы сделать переход от *Sendmail* к Postfix максимально простым. Для отправки сообщения пользователю *root* введите такую команду:

```
echo test | /usr/sbin/sendmail -f root root && tail -f /var/Log/mail
```

Будет отправлен текст *test* пользователю *root* с адресом отправителя *root*, а затем для проверки статуса доставки будет открыт почтовый журнал. С помощью изучения сообщений в журнале можно убедиться в том, что сообщение было доставлено адресату. В этом же можно убедиться, просмотрев файл, в котором хранятся сообщения адресата. Имя файла совпадает с именем адресата. Для того, чтобы узнать, где хранятся такие файлы, необходимо выполнить следующую команду:

```
postconf mail_spool_directory
```

### Отправка сообщения из командной строки с помощью клиента mail

Теперь проверим, сможем ли мы отправить сообщение пользователю *root* из почтового клиента, расположенного на *localhost*. Это второй простейший тест:

```
mail admin
```





Вот краткая инструкция того, как работать с программой *mail*:

1. Введите *mail* в командной строке.
2. Введите имя учетной записи, которой вы хотите адресовать сообщение, и нажмите "ENTER".
3. Получив приглашение, введите тему и нажмите "ENTER".
4. Введите текст сообщения.
5. Для отправки сообщения перейдите на новую строку и введите одну точку ("."), затем нажмите "ENTER".

Для проверки того, отправлено ли сообщение пользователю *admin*, можно использовать следующую команду:

```
less /var/mail/admin
```

### Отправка сообщения из сеанса telnet

Простейший почтовый клиент – это клиент *telnet*, установивший соединение с портом 25, выделенным для SMTP. Мы пойдем таким сложным путем, т. к. хотим исключить побочные эффекты, возможные при использовании более комфортных (и содержащих больше ошибок) почтовых клиентов. Вот как отправляется сообщение с помощью *telnet* (выделенным шрифтом указаны команды, простым – ответы сервера):

```
telnet mail.itstep.org 25
Trying 172.16.0.1...
Connected to mail.itstep.org.
Escape character is '^].
220 mail.itstep.org ESMTP Postfix
HELO client.example.com
250 mail.itstep.org
MAIL FROM: <client@example.com>
250 Ok
RCPT TO: <root@itstep.org>
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Test mail from a telnet session.
.
250 Ok: queued as 69F1A7247
QUIT
221e
```





## 21.2.7 Сопоставление электронных адресов именам пользователей

Давайте представим, что в нашей компании появился новый сотрудник по имени Иван Иванов, и ваша задача – создать для него учетную запись электронной почты. Иван работает в отделе продаж, и предполагается, что он должен получать в один почтовый ящик сообщения, отправленные по адресам `ivan@itstep.org`, `ivan.ivanoff@itstep.org` и `ivanoff@itstep.org`.

Туда же ему должны приходить сообщения, отправленные на адрес отдела `sales@itstep.org`, где он работает вместе с Верой и Надеждой, которые также получают всю почту, приходящую на адрес `sales@itstep.org`. Иван уже получил учетную запись `ivan`, которая дает ему доступ к его файлам.

Вам надо сопоставить все эти синонимы (`ivan@itstep.org`, `sales@itstep.org` и т. д.) его локальному имени пользователя. Это делается с помощью записей в файле `/etc/aliases`. В случае с Иваном, для четырех синонимов достаточно трех записей. Для `ivan@itstep.org` синоним не нужен, поскольку все сообщения на этот адрес будут доставляться Ивану как владельцу учетной записи `ivan`. Надо добавить в файл `/etc/aliases` следующие записи:

```
# users
ivan.ivanoff: ivan
ivanoff: ivan

# groups
sales: vera, nadezhda, ivan
```

В завершение, для обновления файла `aliases.db`, вам нужно будет выполнить команду:

```
postalias hash:/etc/postfix/aliases
```

Или:

```
newaliases
```

Когда вы добавили все необходимые вам синонимы, надо протестировать эти почтовые ящики подобно тому, как вы делали это раньше.

- |          |                  |            |             |             |           |
|----------|------------------|------------|-------------|-------------|-----------|
| ■ Киев   | ■ Днепропетровск | ■ Львов    | ■ Ровно     | ■ Мариуполь | ■ Полтава |
| ■ Одесса | ■ Донецк         | ■ Николаев | ■ Запорожье | ■ Луганск   | ■ Харьков |



## 21.2.8 Установка разрешений на пересылку почты из своей сети

Открытые почтовые серверы (*open relays*) – это ночной кошмар администратора почтового сервера. При любой установке по умолчанию ретрансляция в Postfix ограничена. В типовой конфигурации Postfix будет отправлять только сообщения, полученные от IP адресов собственной сети. Сетевые адреса Postfix получает из настроенных вами на сервере интерфейсов.

На сервере под Linux Postfix будет доверять всем подсетям, в которые входят интерфейсы компьютера. Выполните в Linux команду `ifconfig`, чтобы получить список всех подсетей, которым Postfix будет доверять по умолчанию.

Настройки по умолчанию действуют до тех пор, пока ваш сервер и хосты, которые используют установленный на нем Postfix, находятся в рамках одной подсети. По мере роста или усложнения структуры вашей сети появляется вероятность того, что эти настройки придется изменить. Например, вам может понадобиться запустить Postfix в демилитаризованной зоне (DMZ) в диапазоне IP адресов, отличающемся от диапазона адресов, используемого вашими внутренними хостами. В такой ситуации, вероятно, Postfix не позволит вашим клиентам отправлять почту внешним адресатам, и вам придется изменить конфигурацию, установив полномочия на ретрансляцию.

Расширение или ограничение прав на ретрансляцию может быть групповым – при помощи параметра *mynetworks\_style*, подходящего для вашей сетевой архитектуры, или же индивидуальным – вы вручную указываете список IP адресов или их диапазонов в нотации CIDR (Classless Inter Domain Routing – бесклассовая междоменная маршрутизация) для *mynetworks*.

Оба метода требуют выполнения изменений конфигурации в файле `main.cf` вручную. Такие усилия по администрированию будут разумны для статических диапазонов IP адресов, потому что они редко изменяются. Если же вы хотите разрешить ретрансляцию для хостов с динамическими IP адресами (которые регулярно меняют свой IP адрес), то ручное администрирование нерационально. Ручное внесение изменений вскоре превратится в скучную и утомительную работу. Автоматизировать этот процесс можно с использованием SMTP-аутентификации, которая будет рассмотрена в следующих разделах.



## 21.2.8.1 Групповые полномочия на ретрансляцию

Групповые полномочия на ретрансляцию задаются путем выбора значения параметра *mynetworks\_style: class, subnet или host*.

### class

Если выбрано значение *class*, Postfix распространит полномочия на ретрансляцию почты для всей IP сети класса A/B/C, для которой сконфигурирован сервер. Например, если вы запускаете Postfix на компьютере с IP адресом 192.0.34.166 и устанавливаете значение параметра *mynetworks\_style = class*, то Postfix будет доверять всей сети класса C, 192.0.34.0/24, и разрешит пересылку для хостов этого диапазона.

### subnet

Значение *subnet* указывает, что Postfix разрешит ретрансляцию только для тех подсетей, для которых настроены сетевые интерфейсы сервера. Например, если вы запускаете Postfix на компьютере с IP адресом 192.0.34.166/30 и задаете значение параметра *mynetworks\_style = subnet*, то Postfix будет доверять всем хостам внутри данного диапазона.

### host

При выборе значения *host* Postfix разрешит ретрансляцию только для того сервера, на котором работает Postfix. Например, если вы запускаете Postfix на компьютере с IP адресами 192.0.34.166 и 127.0.0.1 и указываете *mynetworks\_style = host*, то Postfix будет доверять только этим хостам (IP адреса 127.0.0.1 и 192.0.34.166).

## 21.2.8.2 Индивидуальные полномочия на ретрансляцию

Индивидуальные полномочия на ретрансляцию задаются в *mynetworks* путем создания списка всех хостов и сетей, для которых Postfix может пересылать сообщения (используется CIDR нотация, значения разделяются запятыми). Например, если Postfix обслуживает сеть, которая объединяет две подсети (192.168.100.0/24 и 192.168.200.0/24), и вы хотите разрешить серверу ретрансляцию для всех хостов

- |          |                  |            |             |             |           |
|----------|------------------|------------|-------------|-------------|-----------|
| ■ Киев   | ■ Днепропетровск | ■ Львов    | ■ Ровно     | ■ Мариуполь | ■ Полтава |
| ■ Одесса | ■ Донецк         | ■ Николаев | ■ Запорожье | ■ Луганск   | ■ Харьков |



демилитаризованной зоны, в которой он находится (10.0.0.0/30), а также для всех его собственных локальных интерфейсов (127.0.0.0/8), то следует указать такой список:

```
mynetworks = 127.0.0.0/8, 192.168.100.0/24, 192.168.200.0/24, 10.0.0.0/30
```

Если у вас множество IP адресов и диапазонов, то такое перечисление внутри main.cf может стать слишком сложным. Вместо этого вы можете указать путь кциальному файлу (*mynetworks = hash:/etc/postfix/mynetworks*) и создать сложный список в этом файле. Однако в этом файле невозможно использование CIDR нотации для сетей. Если вам необходима CIDR нотация, укажите:

```
mynetworks = cidr:/etc/postfix/mynetworks
```

## 21.3 Использование ограничений на передачу сообщений

Понятие передачи сообщений имеет два аспекта: SMTP соединение, которое обеспечивает передачу, и передаваемые данные (которые обычно называют «электронным письмом» или «сообщением»). Термины, используемые для описания передачи сообщений, не были придуманы на пустом месте; они были позаимствованы у старинной, но хорошо известной и устоявшейся системы, которую люди прошлых столетий называли «почтой».

Когда речь идет об обычной почте, значение терминов «доставщик», «конверт», «заголовок», «тело» и «вложение» хорошо известно. По отношению к электронной почте эти слова являются техническими терминами. В обычной почте доставщик – это почтальон или почтовый курьер. В электронной почте доставщик – это клиент.

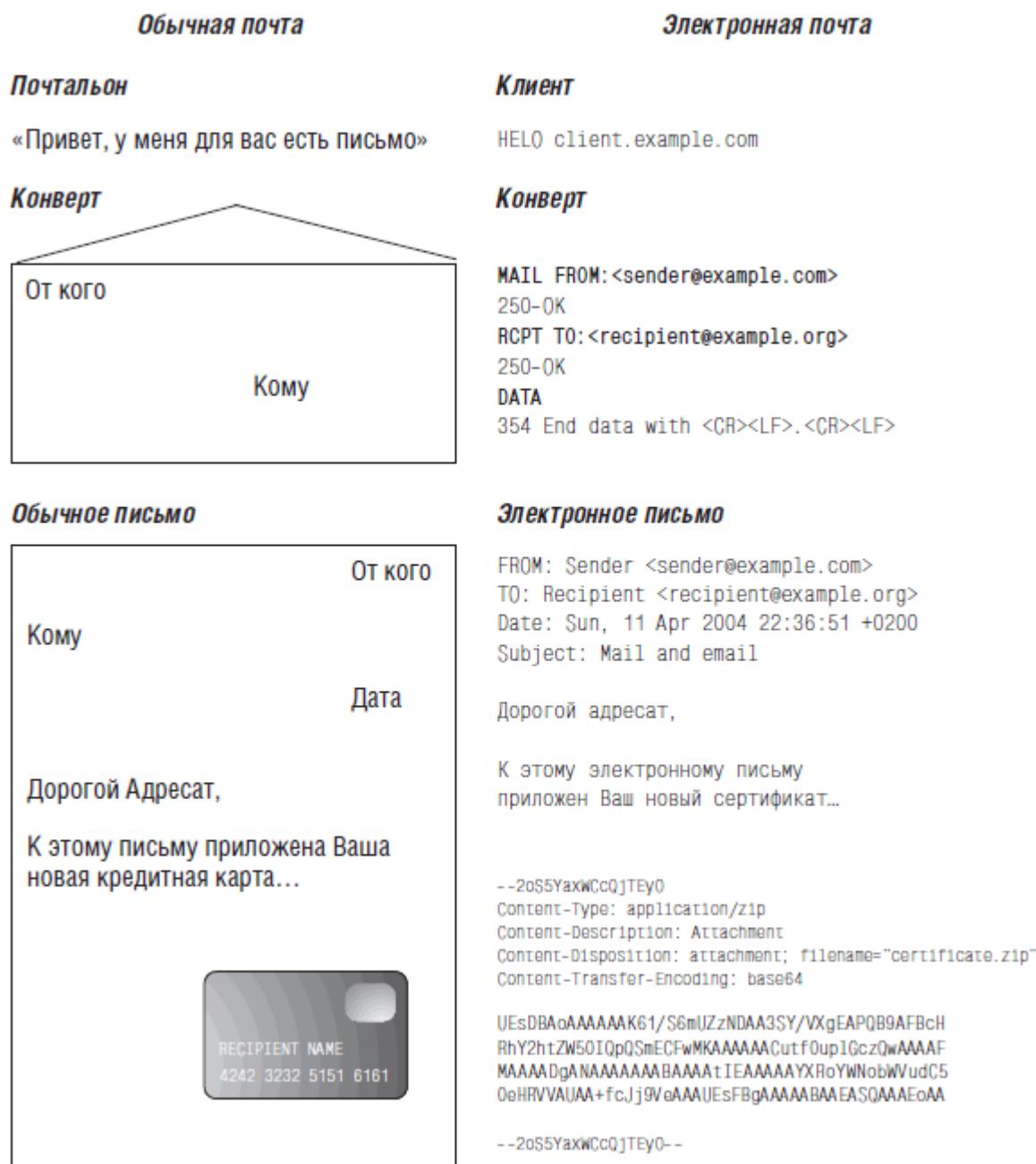
В электронном письме, как и в обычном, конверт служит упаковкой, которая объясняет, как должно быть доставлено содержимое. На конверте указаны отправитель конверта и получатель конверта.

Заголовок предоставляет вам метаданные о сообщении. Как и в обычном письме, это информация об отправителе (заголовок *From:*), получателе (*To:*), дате и времени отправки (*Date:*) и теме сообщения (*Subject:*). Кроме того, заголовки *Received:* электронного сообщения сообщают вам о пути, проделанном сообщением, и времени, в течение которого оно передавалось.





В *теле* электронного сообщения находится собственно его содержимое, совсем как в обычном письме. Если электронное сообщение содержит *вложения*, этот факт будет отмечен в *теле* письма, как это было бы сделано и в обычном письме. *Вложения* не являются обязательными и могут иметь разнообразные форматы. В приведенном ниже изображении производится сопоставление обычного и электронного письма:



Postfix имеет три отдельных группы параметров управления содержимым, которые непосредственно связаны с различными частями сообщения:





### **smtpd\_\*\_restrictions**

Параметры *smtpd\_\*\_restrictions* управляют клиентским соединением и конвертом в процессе передачи сообщений.

### **\*\_checks**

Параметры *\*\_checks* контролируют заголовок, тело и вложения.

### **Фильтры**

Postfix использует фильтры для передачи задач другим (внешним) отбраковывающим приложениям. Фильтры универсальны – они могут контролировать любую частью сообщения, от конверта до вложения.

Каждый из этих параметров имеет большое количество возможных значений; если вы не будете знать, на какую часть сообщения действует каждый из параметров, то ваш контроль содержимого не будет работать.

## **21.3.1 Управление SMTP соединением**

В SMTP соединении участвуют клиент (компьютер, который обращается к SMTP серверу) и передаваемый клиентом конверт. Давайте рассмотрим ситуацию на примере, используя для подключения к серверу программу telnet на вашем компьютере. Начнем с подключения в командной строке к порту 25 вашего почтового сервера:

```
mail:~/Desktop # telnet mail.itstep.org 25
Trying 192.168.40.1...
Connected to mail.itstep.org.
Escape character is '^].
220 mail.itstep.org ESMTP Postfix
```

Возвращаемый сервером код 220 подтверждает имя хоста сервера. Теперь наша очередь представиться серверу:

```
he1o client.itstep.org
250 mail.itstep.org
```

«Рукопожатие» можно осуществить при помощи команды HELO (для SMTP) или EHLO (для ESMTP), указав имя хоста клиента в качестве параметра. В случае успешного





выполнения команды вы получите код возврата 250, за которым будет следовать имя хоста сервера.

Давайте теперь отправим почту. Команда MAIL создает конверт, начиная с указания отправителя. Если сервер признал отправителя, вы снова получите код возврата 250:

```
mail from:sender@itstep.org
250 2.1.0 Ok
```

Следующий этап создания конверта – указание получателя конверта с помощью команды RCPT. Вы можете указать одного или нескольких получателей:

```
rcpt to:recipient@itstep.org
250 2.1.5 Ok
rcpt to:recipient2@itstep.org
250 2.1.5 Ok
```

Для того чтобы отправить настоящее сообщение (включая все дополнительные заголовки, такие как Subject, To и Date), используйте команду DATA:

```
data
354 End data with <CR><LF>.<CR><LF>
subject:test

Message from sender
.
250 2.0.0 Ok: queued as 7B2DBF02C
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

Приведем теперь краткое описание только что рассмотренных понятий, представленное в RFC для электронной почты:

## Клиент

Клиент – это компьютер, отправляющий почту; Postfix запишет имя и IP адрес хоста или же «unknown» (если в результате поиска в DNS невозможно определить имя хоста). Прежде чем будет открыто SMTP соединение, Postfix получает IP адрес клиента из TCP/IP стека ядра, а имя от DNS или из /etc/hosts. Это позволяет Postfix наложить ограничения в случае, если в SMTP соединении возникнет несоответствие IP адреса





клиента и имени хоста. Postfix всегда записывает IP адрес клиента и имя хоста (если оно имеется) в почтовый журнал, а также добавляет эту информацию в последний заголовок сообщения.

## Команда HELO/EHLO

Клиент должен представиться почтовому серверу, сообщив о себе следующие сведения: тип обслуживания и имя хоста. Первая часть команды представления – это запрашиваемый клиентом тип обслуживания. HELO определяет обычное обслуживание, как описано в RFC 821 (<ftp://ftp.rfc-editor.org/in-notes/rfc821.txt>), а EHLO – расширенное, как описано в RFC 2821 (<ftp://ftp.rfc-editor.org/in-notes/rfc2821.txt>). Вслед за запрашиваемым типом обслуживания должна быть идентифицирована личность клиента (предполагается, что клиент предоставит полное имя хоста).

## Конверт

Конверт должен содержать как минимум два различных элемента: ровно одного отправителя конверта и как минимум одного получателя конверта. Клиент отправляет конверт, передавая сначала имя отправителя, затем имена получателей конверта. Если получателей у конверта несколько, то клиент должен передавать их имена одно за другом, каждое с новой строки, ожидая ответа сервера после каждого нового имени. В задачу сервера входит разрешение доставки всем или нескольким получателям.

## Отправитель конверта

Отправитель конверта – это отправитель, которому Postfix ответит в случае ошибки: отложенной доставки или получения уведомления о возврате.

## Получатель конверта

Получатель конверта – это подразумеваемый получатель (получатели) сообщения. Одно сообщение может иметь несколько получателей конверта (например, сообщение нескольким подписчикам списка рассылки). Почтовый сервер требует указания хотя бы одного получателя конверта (в противном случае ему некому доставлять сообщение). Поэтому клиент не может указать пустого получателя конверта (<>).





Практически все данные из предыдущего списка могут быть сфальсифицированы, поэтому Postfix позволяет снизить риск подлога при помощи параметров *smtpd\_\*\_restrictions*, которые определяют ответы на такие вопросы:

1. Откуда пришел клиент?
2. Кем клиент представился?
3. Есть ли у клиента специальные привилегии?
4. Кто является отправителем?
5. Кто является получателем?

Postfix также пытается дать ответы на более сложные вопросы:

1. В правильной ли форме клиент предоставляет информацию Postfix?
2. В правильном ли порядке клиент предоставляет информацию?
3. Всю ли информацию предоставил клиент?
4. Если клиент предоставил не всю необходимую информацию, попытается ли он отправить сообщение?
5. Можно ли определить, корректна ли информация?
6. Если возможно это определить, лжет ли клиент?

Postfix может дать ответы на эти вопросы, исследовав конверт сообщения и элементы SMTP диалога. Когда Postfix отвергает сообщение на основе ограничений для SMTP конверта, он отвергает сообщение до его получения. Поэтому Postfix не отправляет уведомление о «недоставимости» сообщения по адресу отправителя, этим должен заниматься клиент. Это кардинально уменьшает трафик: вместо того чтобы принять сообщение (возможно, с вложением в несколько мегабайт) и затем пытаться отправить обратно сообщение о недоставке, Postfix сразу отказывается принимать сообщение на основании ограничений.

### 21.3.2 Структура электронного сообщения

Сообщение состоит из следующих частей:

1. Заголовки электронного сообщения.
2. Начало тела сообщения.
3. Начало вложения.



Postfix может произвести проверку каждого из этих элементов (*header\_checks*, *body\_checks*, *mime\_header\_checks*) в отдельности. Для того чтобы такие проверки были эффективными, вам необходимо знать, какие обязательные, рекомендованные и необязательные элементы может включать в себя сообщение.

### 21.3.2.1 Заголовки

Заголовок несет в себе метаданные о теле сообщения, такие как кодировка символов и дата отправки. RFC 2822 (<ftp://ftp.rfc-editor.org/in-notes/rfc2822.txt>) делит элементы заголовка на обязательные и рекомендованные. Поля заголовка не обязаны появляться в определенном порядке. Однако рекомендуется отправлять заголовки в следующем порядке: *Return-Path*, *Received*, *Date*, *From*, *Subject*, *Sender*, *To*, *Cc* и т. д. Подробную информацию о заголовках можно найти в документе «Reading Email Headers» (<http://www.stopspam.org/email/headers.html>).

#### Обязательные заголовки

Два элемента заголовка являются обязательными:

##### **Date**

Поле даты обычно содержит дату и время, в которое сообщение было составлено и отправлено. Если клиент отправителя пропустил этот заголовок, то Postfix добавит его.

##### **From**

Это поле идентифицирует личность человека, отправившего сообщение. Если клиент отправителя пропустил этот заголовок, то Postfix добавит его.

#### Рекомендованные заголовки

Приведем перечень рекомендованных элементов заголовка:

##### **Message-Id**

Это поле содержит уникальный идентификатор, который определяет текущую версию текущего сообщения. Клиент генерирует идентификатор сообщения и гарантирует его уникальность. Кроме того, идентификатор сообщения предназначен для компьютеров и необязательно будет означать что-то, понятное для людей. Так как идентификатор сообщения соответствует ровно одному экземпляру конкретного



сообщения, то любые последующие редакции сообщения получают новые идентификаторы. Если клиент отправителя пропустил этот заголовок, то Postfix добавит его.

### To

Это поле определяет основного получателя сообщения. Если клиент отправителя пропустит это поле, то Postfix вставит туда значение параметра конфигурации *undisclosed\_recipients\_header*.

### Subject

Это поле должно содержать очень краткое описание сообщения.

### Cc

Это поле указывает дополнительных получателей сообщения.

### Reply-To

Это поле указывает, куда клиент отправителя должен отправлять ответ на сообщение.

### Content-type

Это поле описано в RFC 1049 (<ftp://ftp.rfc-editor.org/in-notes/rfc1049.txt>), оно определяет структуру тела сообщения.

### MIME-Version

Если такое поле заголовка присутствует, это означает, что тело сообщения было (предположительно) составлено в соответствии с RFC 1521 (<ftp://ftp.rfc-editor.org/in-notes/rfc1521.txt>).

### Received

Каждый транспортный агент, который сталкивается с сообщением, добавляет одну такую строку заголовка для того, чтобы отметить, куда, когда и как поступало сообщение. Сведения из этих полей могут быть полезны при трассировке в случае проблем с передачей.

### Return-Path

Это поле указывает отправителя конверта и используется для определения пути обратно к автору. Почтовый сервер вставляет это поле после доставки локальным агентом доставки, таким как демон local.



## Необязательные заголовки (X-заголовки)

Х-заголовок – это общий термин для дополнительных полей заголовка, имя которых начинается с заглавной буквы «Х», за которой следует дефис. Х-заголовки предназначены только для передачи нестандартных данных, и наоборот – любой нестандартный информативный заголовок должен быть Х-заголовком.

### 21.3.2.2 Тело

В теле помещается собственно сообщение, оно должно находиться после раздела заголовков. Тело может представлять собой открытый или закодированный текст. Тело также может включать вложения, закодированные так, чтобы не допустить искажений при передаче через Интернет (в старые времена многие агенты передачи сообщений не пропускали восемь битов, а отрезание восьмого бита искажает двоичный файл).

### 21.3.2.3 Вложения

Вложения – это файлы, конвертированные в текстовый формат без элементов форматирования (только печатаемые символы), пригодный для отправки в качестве электронного сообщения. В мозаике вложений много различных элементов, о них будет рассказано в последующих подразделах.

#### MIME-кодировки

MIME – это сокращение от Multipurpose Internet Mail Extensions (многоцелевые расширения электронной почты). Речь идет о системе переопределения формата сообщений, описанной в RFC 2045 (<http://www.rfc-editor.org/rfc/rfc2045.txt>). Для двоичных файлов широко используются две MIME-кодировки: *quoted-printable* и *base64*.

#### Quoted-printable

Кодировка quoted-printable предназначена для представления данных, которые по большей части состоят из октетов, соответствующих печатаемым символам в наборе символов US-ASCII. Данные кодируются таким образом, что изменение получившихся октетов в процессе передачи почты маловероятно



## base64

base64 – это система кодирования данных, которая определена в RFC 1421 (<ftp://ftp.rfc-editor.org/in-notes/rfc1421.txt>), а также в еще одном документе: RFC 2045 (<ftp://ftp.rfc-editor.org/in-notes/rfc2045.txt>). Она предназначена для преобразования двоичных данных в печатаемые символы ASCII. По существу это кодировка передачи MIME-содержимого в Интернете, использующая только буквенно-цифровые символы (A–Z, a–z, цифры 0–9) и символы «+» и «/», при этом символ «=» является специальным кодовым суффиксом. Ручное кодирование и декодирование осуществляются при помощи утилит командной строки *tracck*, *tiprasc* и *uudeview*.

Все современные почтовые клиенты поддерживают MIME, и вложения обычно отправляются только в кодировке base64.

## Обработка кодировки

Почтовый клиент занимается кодированием двоичного вложения, а также автоматически создает MIME-структуру, необходимую для того, чтобы встроить текст письма и закодированные вложения в форме, понятной другим поддерживающим MIME почтовым клиентам. Для этого в сообщении должны быть следующие поля:

### MIME-Version

Наличие такого заголовка указывает на то, что сообщение представлено в MIME-формате. Значение, как правило, равно 1.0, так что заголовок обычно выглядит так:

*MIME-Version: 1.0*

### Content-type

Этот заголовок указывает тип и подтип содержимого сообщения, например:

*Content-type: text/plain*

Сочетание типа (*text* в данном примере) и подтипа (*plain*) обычно называется MIME-типов, так что в нашем примере MIME-тип – это *text/plain*. Очень многие форматы файлов имеют зарегистрированные MIME типы. IANA ведет архив всех зарегистрированных типов (<ftp://ftp.isi.edu/in-notes/iana/assignments/media-types>). К





тому же все текстовые типы имеют дополнительный необязательный параметр *charset*, который указывает кодировку символов. Очень большому количеству кодировок символов соответствуют зарегистрированные названия *MIME charset*.

## Типы содержимого

Рассмотрим несколько MIME-типов, которые вам, вероятнее всего, встретятся. В дополнение MIME-тип *multipart-mime-message* (многоэлементное MIME-сообщение) позволяет сообщениям состоять из нескольких различных частей, организованных в древовидную структуру, где узлы-листья имеют не многоэлементный тип содержимого, а узлы, не являющиеся листьями, относятся к одному из многоэлементных типов. MIME-механизм поддерживает (среди прочих) следующие типы:

### **text/plain**

Простые текстовые сообщения используют тип *text/plain*; это значение по умолчанию заголовка *Content-type*.

### **multipart/mixed**

Этот тип указывает текст плюс вложения (*multipart/mixed* с элементом *text/plain* и другими нетекстовыми элементами). MIME-сообщение с вложенным файлом обычно указывает исходное имя файла в заголовке *Content-disposition*, так что тип файла определяется как MIME-типом содержимого, так и расширением имени файла (обычно зависящим от операционной системы). Вирусы часто рассылаются как файлы, в которых заголовки *Content-type* и *Content-disposition* указывают разные типы файлов.

### **message/rfc822**

Это ответ с вложенным исходным сообщением (*multipart/mixed* с элементом *text/plain* и с исходным сообщением в виде элемента *message/rfc822*). Postfix обычно возвращает сообщения таким образом (вложение *message/rfc822* – это исходное сообщение, которое было возвращено).

### **multipart/alternative**

Этот тип определяет содержимое с двумя альтернативными способами просмотра, например сообщение, отправленное как простым текстом, так и в формате HTML (одно





и то же содержимое в форматах text/plain и text/html). Outlook Express использует этот тип содержимого по умолчанию, т. к. отправляет почту одновременно в виде HTML и простого текста.

### 21.3.3 Триггеры ограничений

Ограничения – это мощный инструмент. Чтобы эффективно им пользоваться, необходимо понимать, каким образом происходит SMTP взаимодействие, и какими средствами Postfix это взаимодействие анализирует. Рассмотрим этапы SMTP диалога, обозначенные командами, выдаваемыми клиентом.

Клиент {	\$ telnet mailserver.example.com 25
	220 mailserver.example.com ESMTP Postfix
HELO/EHLO имя хоста {	HELO client.example.com
	250-mailserver.example.com
Отправитель конверта {	MAIL FROM:<sender@example.com>
	250 0k
Получатель(и) конверта {	RCPT TO:<recipient@example.com>
	250 0k
DATA {	DATA
	354 End data with <CR><LF>. <CR><LF>
	From: "Sender" <sender@example.com>
	To: "Recipient" <recipient@example.com>
	Date: Sat, 17 May 2003 15:24:43 +0200
	Here comes the mail content . . .
	.
	250 0k: queued as 0EAFFE1C65
	QUIT
	221 Bye

Каждый новый шаг на приведенном выше рисунке отмечает момент, когда демон smtpd получает очередную порцию информации о клиенте и сообщении, которое он хочет передать.

Postfix использует эти этапы для инициирования триггеров ограничений; для каждого этапа существует свой собственный параметр ограничения, имя которого образовано из названия активного демона, названия этапа и его предназначения.



Поэтому названия триггеров ограничений строятся по такому шаблону: `smtpd_название_этапа_restrictions`. Приведем список всех триггеров ограничений с указанием их поведения по умолчанию:

### **smtpd\_client\_restrictions**

Этот триггер применяется к IP адресу или имени хоста клиента либо к ним обоим. По умолчанию Postfix разрешает подключение любому клиенту.

### **smtpd\_helo\_restrictions**

Этот триггер применяется к аргументу HELO/EHLO клиента и к IP адресу и (или) имени хоста клиента. По умолчанию допускается любой аргумент HELO/EHLO.

### **smtpd\_sender\_restrictions**

Это первый набор триггеров, который относится к частям конверта. Postfix применяет его к отправителю конверта, аргументу HELO/EHLO и клиенту. По умолчанию любому отправителю конверта разрешено отправлять сообщения.

### **smtpd\_recipient\_restrictions**

Этот триггер применяется к получателям конверта, отправителю конверта, аргументу HELO/EHLO и к IP адресу и (или) имени хоста клиента. По умолчанию Postfix допускает любых получателей для клиентов, которые определены в параметре конфигурации `mynetworks`, для остальных же разрешены получатели в доменах из `relay_domains` и `mydomains`. Это не дает Postfix возможности превратиться в открытый почтовый сервер.

### **smtpd\_data\_restrictions**

Этот триггер выявляет клиенты, которые отправляют содержимое письма прежде, чем Postfix ответит на команду DATA. Postfix делает это посредством трассировки команды DATA, когда клиент отправляет команду на сервер. По умолчанию ограничения нет.

### **smtpd\_etrn\_restrictions**





Этот специальный триггер может ограничить клиенты, которые могут запрашивать у Postfix очистку очереди сообщений. По умолчанию всем клиентам разрешено выдавать команду ETRN.

Каждый триггер ограничений соответствует набору ограничений; вы можете воспринимать триггеры как пустые коробки. Для того чтобы от них была какая-то польза, следует вложить в них ограничения.

### 21.3.4 Типы ограничений

В Postfix существуют несколько видов ограничений, которые можно разбить на четыре группы: общие ограничения, переключаемые ограничения, настраиваемые ограничения и дополнительные параметры контроля спама. Рассмотрим по порядку каждый из типов ограничения.

#### ОБЩИЕ ОГРАНИЧЕНИЯ

Ограничения первой группы ничего в SMTP диалоге не проверяют, они просто выполняют команды:

##### **permit**

Разрешает запрос.

##### **defer**

Откладывает запрос.

##### **reject**

Отвергает запрос.

##### **warn\_if\_reject**

Содействует последующим ограничениям; если ограничение после warn\_if\_reject решает отвергнуть запрос, то Postfix на самом деле не отвергает сообщение, а вместо этого пишет в журнал сообщение *reject\_warning*.

##### **reject\_unauth\_pipelining**

Отвергает запрос, когда клиент отправляет команды SMTP раньше времени, еще не зная о том, действительно ли Postfix поддерживает конвейерную обработку команд





ESMTP. Тем самым достигается противодействие программам массовой рассылки, которые некорректно используют конвейерную обработку команд ESMTP для ускорения доставки.

## ПЕРЕКЛЮЧАЕМЫЕ ОГРАНИЧЕНИЯ

Ограничения второго типа работают как переключатели. Их можно включить или выключить и при активации они проверяют выполнение некоторого условия. Приведем их неполный список:

### **smtpd\_helo\_required**

Это ограничение требует от клиентов отправки команды HELO (или EHLO) в начале сеанса SMTP. Наличия команды HELO/EHLO требуют RFC 821 и RFC 2821.

### **strict\_rfc821\_envelopes**

Это ограничение регулирует степень терпимости Postfix к ошибкам в адресах, указанных в команде MAIL FROM или RCPT TO. Применение строгих мер может остановить нежелательную почту, но может также и заблокировать легальные сообщения от плохо написанных клиентов.

### **disable\_vrfy\_command**

SMTP команда VRFY позволяет клиентам проверять существование получателя. Это ограничение позволяет отменить команды VRFY.

### **allow\_percent\_hack**

Это ограничение регулирует преобразование из формы *user%domain* в *user@domain*.

## НАСТРАИВАЕМЫЕ ОГРАНИЧЕНИЯ

Настраиваемые ограничения – это карты, которые работают как фильтры. В каждой записи карты ключ является фильтром, а значение – тем действием, которое необходимо выполнить при совпадении (список возможных действий приведен в разделе «Общие ограничения»).

Рассмотрим несколько видов настраиваемых ограничений:

- |          |                  |            |             |             |           |
|----------|------------------|------------|-------------|-------------|-----------|
| ■ Киев   | ■ Днепропетровск | ■ Львов    | ■ Ровно     | ■ Мариуполь | ■ Полтава |
| ■ Одесса | ■ Донецк         | ■ Николаев | ■ Запорожье | ■ Луганск   | ■ Харьков |





## HELO (EHLO) имя хоста

Эти ограничения относятся к именам хостов, которые клиенты могут отправлять с командой HELO или EHLO.

## Имя хоста/адрес клиента

Этими ограничениями определяются клиенты, которые могут устанавливать SMTP соединения с почтовым сервером.

## Адрес отправителя

Эти ограничения определяют адреса отправителей (конвертов), которые Postfix разрешает для использования в командах MAIL FROM.

## Адрес получателя

Эти ограничения определяют адреса получателей (конвертов), которые Postfix разрешает для использования в командах RCPT TO.

## ETRN-команды

Ограничение накладывается на клиенты, которые могут выдавать команды ETRN.

## Проверка заголовка

Ограничение регулирует заголовки сообщений. Шаблоны применяются ко всему логическому заголовку целиком, даже в том случае, когда логический заголовок занимает несколько физических строк текста.

## Проверка тела

Ограничения накладываются на текст, который может появляться в строках тела сообщения.

## Черные списки DNSBL

Эти черные списки ограничивают соединения от IP адресов (клиентов), которые включены в черные списки DNSBL.

## Черные списки RHSBL

Эти черные списки запрещают те домены отправителей (конверта), которые присутствуют в черных списках RHSBL.



## ДОПОЛНИТЕЛЬНЫЕ ПАРАМЕТРЫ КОНТРОЛЯ СПАМА

Дополнительные параметры контроля спама поддерживают другие ограничения или возможности, не входящие в функциональность Postfix по умолчанию. Приведем лишь несколько таких ограничений:

### **default\_rbl\_reply**

Создает шаблон ответа по умолчанию, который будет использоваться при блокировании запроса SMTP клиента ограничением reject\_rbl\_client или reject\_rhsbl\_sender.

### **permit\_mx\_backup\_networks**

Ограничивает использование функции контроля за пересылкой permit\_mx\_backup теми адресатами, у которых основные хосты MX входят в указанный список сетей.

### **rbl\_reply\_maps**

Определяет таблицы поиска и шаблоны ответов DNSBL, индексированные по имени домена DNSBL. Если шаблон не найден, Postfix будет использовать шаблон default\_rbl\_reply.

### **relay\_domains**

Указывает Postfix на необходимость приема почты для этих доменов, несмотря на то, что данный сервер не является местом их конечного назначения.

### **smtpd\_sender\_login\_maps**

Определяет пользователя, которому разрешено использовать определенный адрес MAIL FROM (отправитель конверта). Для того чтобы Postfix мог использовать это ограничение, ему необходимо знать имя пользователя, так что клиент должен идентифицироваться посредством SMTP аутентификации.

## 21.3.5 Области применения ограничений

Для того чтобы правильно применять ограничения, необходимо понимать, на каком этапе SMTP взаимодействия их можно использовать. Некоторые ограничения не имеют никакого смысла на определенных этапах. В следующей таблице представлено, на каком этапе какие ограничения следует использовать.

Киев	Днепропетровск	Львов	Ровно	Мариуполь	Полтава
Одесса	Донецк	Николаев	Запорожье	Луганск	Харьков





Этап	Ограничение
<i>Клиент (IP адрес и/или имя хоста)</i>	check_client_access reject_rbl_client reject_rhsbl_client reject_unknown_client
<i>Команда HELO/EHLO имя_хоста</i>	check_helo_access permit_naked_ip_address reject_invalid_hostname reject_non_fqdn_hostname reject_unknown_hostname
<i>Отправитель Конверта</i>	check_sender_access reject_non_fqdn_sender reject_rhsbl_sender reject_unknown_sender_domain reject_unverified_sender
<i>Получатель конверта</i>	check_recipient_access permit_auth_destination permit_mx_backup reject_non_fqdn_recipient reject_unauth_destination reject_unknown_recipient_domain reject_unverified_recipient
<i>Команда DATA</i>	reject_unauth_pipelineing

### 21.3.6 Создание ограничений

Ограничения могут стать очень сложными, и вы рискуете бесконечными усложнениями просто испортить свой почтовый сервер, запутавшись в попытках настроить ограничения. При создании ограничений старайтесь придерживаться следующих правил:

- Некорректная запись сделает ваши ограничения бесполезными.
- Имеет значение то, на каком этапе происходит оценка ограничений.
- Имеет значение порядок появления ограничений в триггере. Предыдущие действия влияют на оценку последующих ограничений.





Для записи ограничений можно пользоваться таким шаблоном:

```
триггер_ограничения = условное_ограничение, настраиваемое_ограничение \
maptype:/path/to/the/map, общее_ограничение
```

Одно ограничение может иметь размер, превышающий ширину строки, поэтому можно добавить пробел в начало каждой строки продолжения, чтобы Postfix распознал строки как относящиеся к одному параметру. Кроме того, запятые, разделяющие ограничения, являются необязательными. Так что приведенная ниже запись эквивалентна предыдущей (и гораздо удобнее для чтения):

```
триггер_ограничения =
  условное_ограничение
  настраиваемое_ограничение maptype:/path/to/the/map
  общее_ограничение
```

Вообще говоря, Postfix не осуществляет оценку и выполнение ограничений на основе триггера ограничений сразу же после наступления соответствующего этапа SMTP взаимодействия. Postfix ждет того момента, когда клиент отправит данные первого получателя конверта. Наличие этой задержки объясняется тем, что некоторые почтовые клиенты продолжают пытаться отправить свое сообщение, если сервер отвергает команду прежде, чем закончена передача данных, по крайней мере, одного получателя конверта.

Для того чтобы изменить такое поведение по умолчанию, установите параметр *smtpd\_delay\_reject* в значение *no*. Однако, несмотря на то, что можно отследить такие клиенты и создать список исключений, чтобы гарантировать, что они не будут прерваны, лучше всего подождать завершения всех этапов, и уже затем применять ограничения. Тем самым вы не только уменьшите сложность вашей почтовой системы, но и соберете больше сведений о попытке доставки сообщений.

Для того чтобы понять, как параметр *smtpd\_delay\_reject* влияет на оценку ограничений, посмотрите на рисунок:





If `smtpd_delay_reject = yes ...`

`smtpd_client_restrictions,`  
`smtpd_helo_restrictions,`  
`smtpd_sender_restrictions,`  
`smtpd_recipient_restrictions`

`smtpd_data_restrictions`

```
$ telnet mailserver.example.com 25
220 mailserver.example.com ESMTP Postfix
HELO client.example.com
250-mailserver.example.com
MAIL FROM:<sender@example.com>
250 Ok
RCPT TO:<recipient@example.com>
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
From: "Sender" <sender@example.com>
To: "Recipient" <recipient@example.com>
Date: Sat, 17 May 2003 15:24:43 +0200

Here comes the mail content . . .
.
250 Ok: queued as 0EAFFE1C65
QUIT
221 Bye
```

If `smtpd_delay_reject = no ...`

`smtpd_client_restrictions`  
`smtpd_helo_restrictions`  
`smtpd_sender_restrictions`  
`smtpd_recipient_restrictions`

`smtpd_data_restrictions`

Настраиваемые ограничения используют карты. Когда Postfix находит ключ в карте ограничений, выполняется значение (действие), соответствующее этому ключу. Карта может выглядеть так:

```
10.0.0.1 PERMIT Private IP from VPN transfer tunnel
172.16.0 REJECT Private IP address cannot come from outside
168.100.1.3 DUNNO
192.0.34.166 OK
```

Приведенная карта содержит четыре различных действия для настраиваемых ограничений: PERMIT, REJECT, DUNNO и OK. Эти значения указывают Postfix, как следует поступить с клиентом, отправителем или получателем. Приведем лишь самые распространенные действия:

### OK

Ниаких возражений против клиента и сообщения. Postfix прекращает оценку ограничений в текущем наборе ограничений и переходит к следующему набору.

### PERMIT

Аналог OK.





## REJECT

Незамедлительно отвергает сообщение, игнорируя все последующие ограничения. Сообщение отвергается.

## DUNNO

Прекращает оценку текущего ограничения, но приступает к следующему ограничению текущего набора ограничений.

Порядок ограничений внутри набора очень важен, т. к. первое совпадение, возвращающее OK или REJECT, сразу же останавливает оценку ограничений текущего набора (при этом действие REJECT означает, что клиент, получатель или отправитель безоговорочно отвергается). Postfix читает и применяет ограничения сверху вниз или, если вы пишете их в одну строку, слева направо. Для сложных ограничений проще использовать многострочную форму:

```
smtpd_recipient_restrictions =
check_recipient_access hash:/etc/postfix/recipients_restrictions,
permit_sasl_authenticated,
permit_mynetworks,
reject_unauth_destination,
reject_unauth_pipelining,
reject_rbl_client relays.ordb.org
permit
```

Для любого клиента, вызывающего много ошибок при общении с smtpd (например, инициирующего REJECT в ограничении или приводящего к синтаксической ошибке в аргументах), smtpd делает небольшую паузу, прежде чем принимать последующие команды в том же сеансе. Такое поведение служит защитой от некорректно работающего клиентского программного обеспечения.

Существует ряд настроек, регулирующих подобные случаи. Параметр *smtpd\_error\_sleep\_time* определяет длительность паузы (в секундах) после каждой ошибки (по умолчанию – одна секунда). Параметр *smtpd\_soft\_error\_limit* служит своего рода «замедлителем»: когда удаленный SMTP клиент делает несколько ошибок, SMTP сервер Postfix может добавлять дополнительные задержки перед ответом. Наконец, вы можете прервать сеанс, используя параметр *smtpd\_hard\_error\_limit*.

Эти три параметра работают вместе следующим образом:

- |          |                  |            |             |             |           |
|----------|------------------|------------|-------------|-------------|-----------|
| ■ Киев   | ■ Днепропетровск | ■ Львов    | ■ Ровно     | ■ Мариуполь | ■ Полтава |
| ■ Одесса | ■ Донецк         | ■ Николаев | ■ Запорожье | ■ Луганск   | ■ Харьков |





- Если клиент вызывает ошибки и общее количество ошибок в текущем SMTP сеансе меньше или равно значению параметра `smtpd_soft_error_limit`, то каждая ошибка приводит к задержке на значение `smtpd_error_sleep_time`.
- Если клиент вызывает ошибки и общее количество ошибок в SMTP сеансе превышает значение `smtpd_soft_error_limit`, то каждая ошибка вызывает дополнительную задержку на количество секунд, равное числу ошибок сверх значения параметра `smtpd_soft_error_limit`.
- Если количество ошибок клиента превышает значение `smtpd_hard_error_limit`, то Postfix завершает сеанс.

Например, давайте представим, что параметры определены так:

```
smtpd_soft_error_limit = 5
smtpd_hard_error_limit = 10
smtpd_error_sleep_time = 1s
```

Если клиент вызвал 11 ошибок в одном сеансе, то Postfix делает паузы на 1, 1, 1, 1, 1, 2, 3, 4, 5, и 6 секунд соответственно, а после 11-й ошибки сервер отключается.

Класс ограничений – это специальная форма триггера ограничений, который не предопределен и не связан ни с каким конкретным этапом SMTP взаимодействия. Вы определяете их по мере необходимости и инициируете, ссылаясь на них в картах настраиваемых ограничений. Пусть, например, у вас есть карта настраиваемого ограничения для проверки адресов отправителей конвертов, и вы хотите инициировать другой набор ограничений в случае, если отправитель конверта соответствует `example.com`. В этом случае вы можете поместить этот новый набор ограничений в новый класс с именем `check_if_example.com_sender`.

Сначала объявляем новый класс в файле `main.cf`.

```
smtpd_restriction_classes = check_if_example.com_sender
```

Теперь, также внутри `main.cf`, добавляем в новый класс несколько ограничений:

```
check_if_example.com_sender =
check_sender_access hash:/etc/postfix/bounces
check_sender_access hash:/etc/postfix/valid_example.com_senders
check_sender_access regexp:/etc/postfix/nice_reject regexp
```





Эти новые ограничения анализируют отправителя конверта (но здесь могли бы использоваться любые ограничения, подходящие для текущего этапа SMTP диалога). Не беспокойтесь о картах для данных ограничений – далее вы увидите, как их определить. Однако нам до сих пор не хватает чего-то важного... Как инициировать `check_if_example.com_sender`?

Для этого необходимо наличие ограничения `check_sender_access` в наборе `smtpd_*_restrictions`. Давайте будем считать, что у вас уже есть такой набор ограничений с картой, которая разрешает отправителей из `foo.com` и отвергает отправителей из `bar.org` (возможные действия описаны в разделе «Влияние действий на оценку ограничений» ранее в этой главе):

```
foo.com OK
bar.org REJECT
```

Для добавления нового класса ограничений расширьте карту следующим образом:

```
foo.com OK
bar.org REJECT
example.com check_if_example.com_sender
```

Как видите, главное в использовании классов ограничений – это найти правильное место для их вставки в карту настраиваемого ограничения.

### 21.3.7 Ограничения по умолчанию

Postfix поставляется с надежным набором ограничений по умолчанию, которые не дают вашему компьютеру превратиться в открытый почтовый сервер. Для того чтобы познакомиться с ограничениями по умолчанию, необходимо попросить `postconf` вывести эти значения для `smtpd_recipient_restrictions`:

```
mail:~/Desktop # postconf -d smtpd_recipient_restrictions
smtpd_recipient_restrictions = permit_mynetworks, reject_unauth_destination
```

Postfix оценивает ограничения в том порядке, в котором они приведены. В данном случае, если клиент хочет переслать сообщение, Postfix проверяет, исходит ли соединение от хоста, указанного в `mynetworks`. Если это так (ограничение `permit_mynetworks` оценивается как OK), то Postfix принимает сообщение для доставки.





Если клиент не относится к сети из *mynetworks*, Postfix оценивает *reject\_unauth\_destination*. Это ограничение отклоняет попытки пересылки, проверяя, относится ли получатель сообщения к доменам места назначения и доменам пересылки, указанным в ваших настройках.

Если получатель не относится к этим доменам, то *reject\_unauth\_destination* возвращает REJECT, и Postfix сообщает клиенту о том, что пересылка невозможна. Если место назначения сообщения входит в зону ответственности Postfix, ограничение *reject\_unauth\_destination* возвращает DUNNO, и Postfix переходит к оценке следующего ограничения. Если же в списке больше нет ограничений, то Postfix считает, что по умолчанию подразумевается permit, и принимает сообщение. Эти два ограничения – та основа, которая защищает сервер от превращения в открытый почтовый сервер, но они не защищают пользователей от спама и не заставляют клиенты вести себя корректно.

### 21.3.8 Требование соответствия RFC

Требование надлежащего поведения (в соответствии с RFC) от локальных и удаленных клиентов – это первый шаг к тому, чтобы управляемый вами корабль не дал течи. Это не только гарантирует, что ваш почтовый сервер передает другим почтовым серверам корректные сообщения, но и заставляет удаленные клиенты вести себя правильно.

Такое требование полезно для защиты от спамеров, которые всегда спешат, не следуют правилам и фальсифицируют идентификационную информацию. В этом разделе будет показано, как накладывать ограничения на имя хоста, отправителя и получателя конверта, чтобы добиться соответствия RFC.

#### 21.3.8.1 Ограничение на имя хоста в команде HELO/EHLO

Хорошой отправной точкой будет требование Postfix к клиентам относительно их корректного приветствия, если они хотят, чтобы их сообщения были отправлены на ваш сервер или переданы через него. Существует целый ряд ограничений, которые могут быть наложены на HELO/EHLO-часть SMTP диалога, начиная с простого требования отправки имени хоста до требования отправки достоверного имени хоста.

#### Требование указания имени хоста





Параметр *smtpd\_helo\_required* требует, чтобы все клиенты, открывающие SMTP соединение, выполняли команду HELO или EHLO. Такого обязательного приветствия требует как RFC 821 (<ftp://ftp.rfc-editor.org/in-notes/rfc821.txt>), так и RFC 2821 (<ftp://ftp.rfc-editor.org/in-notes/rfc2821.txt>), но Postfix по умолчанию устанавливает этот параметр в значение no. Для того чтобы включить данное требование, добавьте в файл *main.cf* такую строку:

```
smtpd_helo_required = yes
```

После перезагрузки конфигурации Postfix будет отклонять сообщения любого клиента, который не представится должным образом. Для того чтобы проверить это, подключитесь к вашему серверу и попробуйте инициировать передачу сообщения без команды HELO. Вот как отреагирует Postfix, требующий указания имени хоста:

```
mail:~/Desktop # telnet mail.itstep.org 25
Trying 192.168.40.1...
Connected to mail.itstep.org.
Escape character is '^].
220 mail.itstep.org ESMTP Postfix
mail from:sender@itstep.org
503 5.5.1 Error: send HELO/EHLO first
```

### Требование указания полностью определенного доменного имени хоста

Команда HELO/EHLO – это, конечно, хорошо, но клиенты также должны передать вместе с приветствием полное имя хоста (например, HELO client.example.com). Более того, RFC требуют, чтобы указывалось полностью определенное доменное имя хоста (FQDN).

Postfix будет отвергать сообщения от любого клиента, не предоставившего полностью определенное доменное имя хоста, если вы установите параметр *reject\_non\_fqdn\_hostname* внутри ограничения *smtpd\_recipient\_restrictions*.

Когда вы добавите параметр *reject\_non\_fqdn\_hostname* в список *smtpd\_client\_restrictions*, он будет иметь такой вид в файле *main.cf*:

```
smtpd_client_restrictions = reject_non_fqdn_hostname
```

- |          |                  |            |             |             |           |
|----------|------------------|------------|-------------|-------------|-----------|
| ■ Киев   | ■ Днепропетровск | ■ Львов    | ■ Ровно     | ■ Мариуполь | ■ Полтава |
| ■ Одесса | ■ Донецк         | ■ Николаев | ■ Запорожье | ■ Луганск   | ■ Харьков |





```
mail:~/Desktop # telnet mail.itstep.org 25
Trying 192.168.40.1...
Connected to mail.itstep.org.
Escape character is '^].
220 mail.itstep.org ESMTP Postfix
heLo test
250 mail.itstep.org
mail from:a@itstep.org
250 2.1.0 Ok
rcpt to:sender@itstep.org
504 5.5.2 <test>: Helo command rejected: need fully-qualified hostname
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

## Требования к символам, составляющим имя хоста

RFC определяет, что не только имена хостов, отправляемые с командой HELO/EHLO, должны быть полностью определенными доменными именами, но и используемые для составления таких имен символы должны подчиняться требованиям к построению имени хоста. Корректное доменное имя должно включать в себя как минимум следующие элементы:

- Домен верхнего уровня, такой как com
- Имя домена, например example
- Точку (.), разделяющую домен верхнего уровня и доменное имя

Любое другое имя хоста, скорее всего, не будет корректно разрешено, что затруднит (или даже сделает невозможным) взаимодействие между сервером и клиентом. Используя параметр *reject\_invalid\_hostname* в списке *smtpd\_client\_restrictions*, вы можете указать Postfix, что не следует общаться с такими клиентами. Вот пример, показывающий, куда можно вставить этот параметр:

```
smtpd_client_restrictions = reject_invalid_hostname
```

Как и прежде, проверяем ограничение, подключаясь к своему почтовому серверу от удаленного хоста и предоставляем недействительное имя хоста. В данном примере клиент представляется как «..»:





```
mail:~/Desktop # telnet mail.itstep.org 25
Trying 192.168.40.1...
Connected to mail.itstep.org.
Escape character is '^].
220 mail.itstep.org ESMTP Postfix
heLo .
250 mail.itstep.org
mail from:a@itstep.org
250 2.1.0 Ok
rcpt to:sender@itstep.org
501 5.5.2 <.>: HeLo command rejected: Invalid name
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

### 21.3.8.2      Ограничение на отправителя конверта

Доменная часть имени отправителя конверта должна содержать полностью определенное доменное имя (FQDN), и конверт должен принадлежать к существующему домену. Такие отправители конверта, как *sender* и *sender@example*, не указывают полностью определенное доменное имя. Примером полного имени отправителя конверта может быть *sender@example.com*. Неполные адреса могут вызвать путаницу, т. к. адрес отправителя выглядит так, как будто сообщение создано на данном сервере. Возможны два варианта нежелательного развития событий:

- Агент передачи сообщений, который должен возвратить сообщение с неполным именем отправителя конверта, будет возвращать его локальным пользователям. Возврат не дойдет до исходного отправителя.
- Postfix может попытаться «исправить» неправильный адрес, что только усложнит дело. Так как Postfix знает, что отправитель конверта должен иметь полностью определенное доменное имя, он запустит демон *trivial-rewrite* для канонизации адреса путем добавления *\$myorigin* для отправителя *sender* (в результате получится *sender@\$myorigin*) и *\$mydomain* – для *sender@example* (получится *sender@example.\$mydomain*). Следовательно, отправитель конверта для сообщений, полученных от удаленного сервера, будет абсолютно неверным.

Для того чтобы избежать подобных ситуаций, добавьте параметр *reject\_non\_fqdn\_sender* в список *smtpd\_recipient\_restrictions*, например:

```
smtpd_recipient_restrictions =
    reject_non_fqdn_sender
```





```
permit_mynetworks
reject_unauth_destination
permit
```

Проверьте ограничение, подключившись с удаленной машины к вашему почтовому серверу и указав некорректное имя отправителя конверта. Покажем, как ограничения заставят Postfix отклонить сообщения такого отправителя:

```
mail:~/Desktop # telnet mail.itstep.org 25
Trying 192.168.40.1...
Connected to mail.itstep.org.
Escape character is '^].
220 mail.itstep.org ESMTP Postfix
he1o client
250 mail.itstep.org
mail from:sender
250 2.1.0 Ok
rcpt to:recipient@itstep.org
504 5.5.2 <sender>: Sender address rejected: need fully-qualified address
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

## Сообщение из несуществующих доменов

Ответственный почтовый сервер не принимает сообщения от получателей, домены которых не существуют, т. к. в случае невозможности доставки сообщения он не сможет сообщить об этом отправителю. В других конфигурациях возникает двойной возврат, как только агент передачи сообщений пытается уведомить отправителя, и, в конце концов, сообщение с несуществующим доменом отправителя окажется в почтовом ящике администратора почтовой системы.

Для защиты получателей и администраторов почтовой системы от двойного возврата и неправильно составленных сообщений добавьте параметр *reject\_unknown\_sender\_domain* в список *smtpd\_recipient\_restrictions*, например:

```
smtpd_recipient_restrictions =
    reject_unknown_sender_domain
    permit_mynetworks
    reject_unauth_destination
    permit
```





Следующий пример показывает, как можно проверить ограничение (ищем код ошибки 450, который Postfix отправляет в качестве ответа команде MAIL FROM):

```
mail:~/Desktop # telnet mail.itstep.org 25
Trying 192.168.40.1...
Connected to mail.itstep.org.
Escape character is '^].
220 mail.itstep.org ESMTP Postfix
he1o client
250 mail.itstep.org
mail from:sender@invalid.domain
250 2.1.0 Ok
rcpt to:recipient@itstep.org
450 4.1.8 <sender@invalid.domain>: Sender address rejected: Domain not found
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

### 21.3.8.3      Ограничение на получателя конверта

В качестве последнего действия по приведению входящих соединений в соответствие RFC вы можете отвергать сообщения, в которых для получателя конверта указан несуществующий домен или пользователь. Почтовый сервер не должен принимать никакие сообщения для несуществующего домена, т. к. их невозможно доставить. Если почтовый сервер примет сообщение, а затем вернет его, то пользователь может подумать, что у сервера какие-то проблемы, ведь изначально сообщение было принято.

Если настроить почтовый сервер так, чтобы сообщения в несуществующие домены отклонялись, то ваши проблемы станут проблемами того клиента или пользователя, которые отправили сообщение. Для введения такого ограничения используйте параметр *reject\_unknown\_recipient\_domain* в списке *smtpd\_recipient\_restrictions*, например:

```
smtpd_recipient_restrictions =
    reject_unknown_recipient_domain
    permit_mynetworks
    reject_unauth_destination
    permit
```

Как обычно, вы можете протестировать ограничение, отправив письмо в несуществующий домен получателя при ручном подключении к серверу. Покажем на



примере, как Postfix отвергает сообщение из-за того, что домен *invalid.domain* не существует:

```
mail:~/Desktop # telnet mail.itstep.org 25
Trying 192.168.40.1...
Connected to mail.itstep.org.
Escape character is '^].
220 mail.itstep.org ESMTP Postfix
he1o client
250 mail.itstep.org
mail from:sender@itstep.org
250 2.1.0 Ok
rcpt to:rcpt@invalid.domain
450 4.1.2 <rcpt@invalid.domain>: Recipient address rejected: Domain not found
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

## Сообщения неизвестным получателям

Вы можете настроить Postfix так, чтобы сообщения для неизвестного пользователя из вашего домена доставлялись администратору почтовой системы. На первый взгляд такая идея кажется весьма удачной, т. к. администратор может изучить сообщение и при наличии возможности доставить его вручную.

Но несмотря на то, что в теории подобная конфигурация могла бы обеспечить идеальное обслуживание клиентов, использование адресата по умолчанию может привести к DoS атакам (Denial-of-service – отказ в обслуживании) на ваш сервер, как только он станет целью для спамера или червя, использующего атаку по словарю. Такая атака заключается в попытке отправить сообщение существующим получателям за счет рассылки сообщений по адресам, составленным из всех возможных сочетаний букв. Например, атакующий может начать с адреса *aa@yourdomain.com*, затем попробовать *ab@yourdomain.com* и так пройти через все двухбуквенные комбинации вплоть до *zz@yourdomain.com*.

Дело не только в сложности отделения законных сообщений от сообщений, созданных в процессе подобной атаки, но и в том, что сервер подвергается риску в связи с исчерпанием пропускной способности канала, производительности процессора, памяти и дискового пространства, и в результате сдается и останавливает обслуживание запросов на передачу сообщений. Например, вирус Sobig.F привел к перегрузке многих почтовых серверов в августе 2003 года.





Помните, что для Postfix приоритетом является надежность обслуживания. Надежность подразумевает согласованность, и поэтому сервер отвергает почту, адресованную неизвестным пользователям, по умолчанию без какого-либо ручного вмешательства. Это замечательно для автономного сервера Postfix, но полезно и для сервера Postfix, работающего на интеллектуальном хосте, который защищает остальные почтовые серверы.

Postfix определяет корректность адресов получателей, сверяясь с картами. Существуют два параметра конфигурации, которые указывают Postfix, где следует искать такую информацию: *local\_recipient\_maps* и *relay\_recipient\_maps*. В обоих параметрах указывается одна или несколько карт, содержащих действительных получателей.

Параметр *local\_recipient\_maps* определяет действительных локальных получателей, как показано в примере, где получатели определены в файле паролей UNIX и картах псевдонимов:

```
mail:~/Desktop # postconf -d local_recipient_maps
local_recipient_maps = proxy:unix:passwd.byname $alias_maps
```

В то же время параметр *relay\_recipient\_maps* определяет получателей, для которых Postfix пересылает сообщения к конечному месту назначения (серверу почтовых ящиков).

Однако использование *luser\_relay* отменяет параметр *local\_recipient\_maps*, т. к. делает действительными всех локальных получателей. Аналогично запись с групповым именем *catchall* в списке *virtual\_alias\_maps* отменяет отклонение почты, адресованной несуществующим получателям, т. к. групповое имя делает действительными всех получателей. Например, следующая запись карты делает действительными всех получателей в домене example.com:

```
@example.com catchall@localhost
```

## Сообщения получателям с неполным именем

Не полностью указанный адрес, такой как *recipient*, содержит только локальную часть электронного адреса. С приемом таких сообщений для локальных пользователей на компьютере, получающем почту только для одного домена, проблем нет, но





трудности появляются, если ваш почтовый сервер получает сообщения и для других доменов.

Если указана одна лишь локальная часть, то остается слишком много простора для интерпретации почтового адреса. Пусть, например, вы работаете Интернет-провайдером для двух конкурирующих компаний, *example.com* и *example.net*. Если вы получите сообщение для получателя *sales*, куда оно будет отправлено? По какому адресу оно должно попасть – *sales@example.com* или *sales@example.net*, если один и тот же сервер обслуживает оба этих электронных адреса?

В силу вышесказанного вам следует отклонять сообщения с неполными адресами. Не принимайте на себя чужую ответственность. Подготовка сообщения для корректной доставки – это задача отправителя, и он должен определить получателя уникальным способом.

Существует всего одно исключение: вы должны принимать сообщения для *postmaster* при неполноты указанном адресе. На адрес *postmaster* не накладываются вообще никакие ограничения, действующие для получателей (включая ограничения на отправителя, команду *helo* и клиента).

Postfix будет отвергать сообщения для получателей с неполным именем, если вы добавите параметр *reject\_non\_fqdn\_recipient* в ваш список *smtpd\_recipient\_restrictions*, как в следующем примере:

```
smtpd_recipient_restrictions =  
    reject_non_fqdn_recipient  
    reject_unknown_recipient_domain  
    permit_mynetworks  
    reject_unauth_destination  
    permit
```

Проверим ограничение, подключившись к своему почтовому серверу с удаленного компьютера и отправив сообщение с неполным адресом получателя. Для проверки работы ограничения достаточно будет такого сеанса:





```
mail:~/Desktop # telnet mail.itstep.org 25
Trying 192.168.40.1...
Connected to mail.itstep.org.
Escape character is '^].
220 mail.itstep.org ESMTP Postfix
he1o client
250 mail.itstep.org
mail from:sender@itstep.org
250 2.1.0 Ok
rcpt to:recipient
504 5.5.2 <recipient>: Recipient address rejected: need fully-qualified address
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

### 21.3.9 Обеспечение соответствия RFC

Возможно, вы уже обратили внимание на то, что ограничения могут стать довольно сложными. И чем сложнее становятся ограничения, тем выше вероятность того, что среди них появится такое, которое приведет к неправильной работе вашей почтовой системы (или вообще сделает ее абсолютно бесполезной), отвергая сообщения, которые должны быть приняты при любых обстоятельствах. Необходимо избегать непредумышленной блокировки некоторых или всех отправителей. Это важно, т. к. вы можете случайно исключить отправителей, которые могли бы сообщить о недостатках вашей конфигурации.

#### Пустое имя отправителя конверта

Во-первых, никогда не блокируйте пустого отправителя конверта (<>). Этот адрес принадлежит MAILER-DAEMON, почтовый сервер использует его при отправке возвратов и уведомлений о состоянии. Если заблокировать этот адрес, то удаленные серверы не смогут сообщить вашим пользователям о том, что с отправленными ими сообщениями возникли проблемы.

Черные списки, такие как dsn.rfc-ignorant.org, приводят перечень почтовых серверов, которые категорически отказываются принимать почту от отправителей конвертов с пустым именем, так что использующие эти черные списки почтовые серверы не принимают почту от перечисленных там серверов (мы вернемся к этому вопросу в разделе «Отказ доменам отправителей из черных списков»).





Все, что вам нужно, – это рассматривать пустой адрес отправителя конверта как любой другой допустимый адрес и создать хорошие ограничения (противодействующие спаму) для защиты ваших получателей. Пусть ограничения выполняют свою работу, и если вы получите сообщение с пустым именем отправителя, примените его. В конце концов, любой адрес отправителя может оказаться подделкой...

## Специальные учетные записи

На почтовом сервере имеются два адреса, для которых вы всегда должны принимать сообщения; они необходимы для того, чтобы работа почтового сервера соответствовала RFC:

### **postmaster**

Всегда принимайте почту, адресованную администратору почтовой системы postmaster, – это центр обработки информации для вопросов, связанных с электронными сообщениями. Пользователи должны иметь возможность обратиться к администратору за помощью (см. RFC 2821 по адресу <http://www.rfc-editor.org/rfc/rfc2821.txt>).

### **abuse**

Прием почты для адресата abuse гарантирует, что пользователи смогут уведомить вас о возможных почтовых злоупотреблениях, исходящих от вашего сервера (см. RFC 2142 по адресу <http://www.rfc-editor.org/rfc/rfc2142.txt>). Дополнительно (но не обязательно) вы можете принимать сообщения для следующих адресатов, если поддерживаете соответствующие серверы (см. RFC 2142; <http://www.rfc+editor.org/rfc/rfc2142.txt>):

### **webmaster**

Принимайте почту для webmaster, если у вас работает веб-сервер.

### **hostmaster**

Принимайте почту для hostmaster, если у вас работает сервер имен.

Вы можете настроить прием сообщений для этих получателей, используя параметр `check_recipient_access` в сочетании с картой, такой как `/etc/postfix/roleaccount_exceptions`, где перечислены получатели, сообщения для которых должны быть приняты. Такая карта может выглядеть следующим образом

■ Киев	■ Днепропетровск	■ Львов	■ Ровно	■ Мариуполь	■ Полтава
■ Одесса	■ Донецк	■ Николаев	■ Запорожье	■ Луганск	■ Харьков



(значение OK для каждого ключа карты сообщает Postfix о том, что следует принимать сообщения для данного получателя без учета ограничений для получателей):

```
# addresses that you must always accept
postmaster@ OK
abuse@ OK
# addresses that you should accept if you run DNS and WWW servers
hostmaster@ OK
webmaster@ OK
```

После создания этого файла преобразуйте его в карту следующей командой:

```
postmap hash:/etc/postfix/roleaccount_exceptions
```

Затем укажите карту в качестве значения параметра *check\_recipient\_access* в списке ограничений файла *main.cf*, например:

```
smtpd_recipient_restrictions =
reject_non_fqdn_recipient
reject_non_fqdn_sender
reject_unknown_sender_domain
reject_unknown_recipient_domain
permit_mynetworks
reject_unauth_destination
check_recipient_access hash:/etc/postfix/roleaccount_exceptions
permit
```

После перезагрузки конфигурации вы можете спокойно переходить к созданию более сложных правил. Карта с исключениями запрашивается после того, как Postfix проводит проверки на неавторизованную пересылку, так что использование адреса *postmaster@* будет безопасным.

### 21.3.10 Порядок обработки RFC ограничений

Параметры ограничений могут влиять один на другой и мешать друг другу, если будут указаны не в нужном порядке. Давайте, например, посмотрим на такой список:

```
smtpd_recipient_restrictions =
reject_non_fqdn_recipient
reject_non_fqdn_sender
reject_unknown_sender_domain
```





```
reject_unknown_recipient_domain
permit_mynetworks
reject_unauth_destination
check_recipient_access hash:/etc/postfix/roleaccount_exceptions
permit
```

Параметр *permit\_mynetworks* обозначает важную границу между клиентами вашей внутренней сети и внешними клиентами. Параметры, заданные выше этого элемента (включая его самого), относятся как к внутренним, так и к внешним клиентам, в то время как параметры ниже *permit\_mynetworks* применяются только к внешним клиентам.

Параметры, предшествующие *permit\_mynetworks*, требуют базового соблюдения RFC от всех клиентов как внутри вашей сети, так и вне ее. Параметр *reject\_unauth\_destination* не дает вашему серверу превратиться в открытый почтовый сервер. Лучше не указывать никакие параметры, разрешающие пересылку сообщений, пока не задан параметр *permit\_mynetworks*. Далее как можно раньше следует указать *reject\_unauth\_destination*, чтобы быть уверенным в том, что неавторизованный хост никоим образом не сможет использовать ваш сервер как открытый почтовый сервер.

Проверка на SMTP-аутентификацию должна находиться между *reject\_unauth\_destination* и *permit\_mynetworks*. Затем, прежде чем указывать еще какие-то параметры отказа от сообщений, используйте параметр *check\_recipient\_access* для разрешения безусловной доставки специальным почтовым ящикам вашей системы.

И, наконец, отразив все возможные попытки фальсификаций со стороны клиентов, вы можете принимать сообщения, используя параметр *permit*.

### 21.3.11 Порядок обработки RFC ограничений

Спамерам необходимо замаскировать источник своих сообщений, если они не хотят судебного преследования. Обычно они подделывают адрес отправителя конверта или пытаются усыпить бдительность принимающего сервера, сообщая ему, что их клиенту можно доверять – как будто он принадлежит к локальной сети. Ограничения могут проверить и отклонить такие сообщения. Более того, они могут запрашивать черные списки, в которых перечислены спамеры и другие адресаты, сообщения от которых вы



не хотите принимать. В этом разделе будет показано, как провести такие ограничения в жизнь.

### 21.3.11.1 Предотвращение явных фальсификаций

Некоторые спам-программы пытаются скрыть источник сообщения, используя имя хоста вашего почтового сервера как свое собственное в приветствии HELO/EHLO. Postfix воспринимает ситуацию как парадоксальную, ведь единственный хост, который может использовать имя хоста сервера, – это сам сервер. Однако Postfix никогда не стал бы подключаться к своему демону smtpd для отправки почты самому себе, если только не была сделана ошибка конфигурации, приведшая к возникновению петли.

Добавление ограничений после строки `permit_mynetworks` сделает их применимыми только к внешним клиентам, но не к прокси-фильтрам или локальным клиентам с неполной реализацией SMTP. Поэтому вы можете отказывать в SMTP соединении любому клиенту, который приветствует ваш почтовый сервер с именем хоста этого сервера.

Для этого сначала создайте файл карты с именем `/etc/postfix/helo_checks`, содержащий различные вариации имени вашего хоста. Приведем несколько примеров для имени хоста, IP адреса хоста и IP адреса в скобках, которые не должны использоваться внешними клиентами:

```
/^mail\.example\.com$/      REJECT Don't use my hostname
/^192\.0\.34\.166$/          REJECT Don't use my IP address
/^\[192\.0\.34\.166\]$/       REJECT Don't use my IP address
```

Документ RFC 2821 указывает, что сам по себе IP адрес не является разрешенным аргументом для команды HELO. IP адрес разрешен, если он задан в форме `[ipv4address]` (в квадратных скобках) или как IPv6 адрес, `[ipv6:ipv6address]`, опять-таки в квадратных скобках. Чтобы соблюсти все формальности и отказать в обслуживании клиентам, которые отправляют IP адрес без квадратных скобок, добавьте такую строку:

```
/^[\d.]+$/      REJECT Your client is not RFC 2821 compliant
```

Для того чтобы привести карту в действие, укажите ее (и ее тип) как аргумент для параметра `check_helo_access` в списке `smtpd_recipient_restrictions`, например:





```
smtpd_recipient_restrictions =
reject_non_fqdn_recipient
reject_non_fqdn_sender
reject_unknown_sender_domain
reject_unknown_recipient_domain
permit_mynetworks
reject_unauth_destination
check_recipient_access hash:/etc/postfix/roleaccount_exceptions
check_helo_access pcre:/etc/postfix/helo_checks
permit
```

Для проверки ограничения подключитесь к своему почтовому серверу, и укажите собственное имя в приветствии HELO. Вы должны получить отказ, как показано в примере:

```
mail:~/Desktop # telnet mail.itstep.org 25
Trying 192.168.40.1...
Connected to mail.itstep.org.
Escape character is '^].
220 mail.itstep.org ESMTP Postfix
helo mail.itstep.org
250 mail.itstep.org
mail from:a@mail.ru
250 2.1.0 Ok
rcpt to:root@itstep.org
554 5.7.1 <mail.itstep.org>: Helo command rejected: 550 Don't use my hostname
```

### 21.3.11.2 Фиктивные записи сервера имен

Postfix может отвергать сообщения, если очевидно, что записи сервера имен для домена HELO, доменов отправителя и получателя отсутствуют или не позволяют обеспечить корректную передачу сообщения. Приведем ряд обстоятельств, которые могут показаться подозрительными в записях DNS:

#### Фальшивые сети

Некоторые почтовые серверы утверждают, что относятся к сетям, недостижимым для Postfix, в том числе неиспользуемым вами частным сетям (см. RFC 1918, [ftp://ftp.rfc-editor.org/in-notes/rfc1918.txt](http://ftp.rfc-editor.org/in-notes/rfc1918.txt)), сети обратной связи (loopback), широковещательным сетям или сетям многоадресной рассылки.

#### Пристанища спамеров

- Киев ■ Днепропетровск ■ Львов ■ Ровно ■ Мариуполь ■ Полтава
- Одесса ■ Донецк ■ Николаев ■ Запорожье ■ Луганск ■ Харьков





Пристанища спамеров (spam havens) – это сети, про которые известно, что они принадлежат спамерам или предоставляют услуги спамерам. Можно отклонять все сообщения из таких доменов. Информацию о пристанищах спамеров и их деятельности вы можете найти в ROKSO (Register of Known Spam Operations – список известных спамеров, <http://www.spamhaus.org/rokso/index.lasso>).

## «Безразличные» агенты передачи сообщений

«Безразличные» (wildcard) агенты передачи сообщений заявляют, что они ответственны за все домены, даже за несуществующие. Казалось бы, это не должно стать проблемой, ведь вы можете отказать в доступе в случае неизвестных доменов получателя и отправителя.

К сожалению, некоторые регистраторы доменов прониклись идеей перенаправлять неизвестные доменные имена в свой собственный домен. В результате неизвестные домены получают действительную А-запись, что делает бесполезными параметры ограничений *reject\_unknown\_sender\_domain* и *reject\_unknown\_recipient\_domain*.

Во всех описанных случаях используются либо фальшивые записи сервера, либо поддержка спамеров. Для отклонения почты из таких доменов и сетей вы можете создать в файле */etc/postfix/bogus\_mx* карту, содержащую IP адреса вместе с типом ответа, который вы хотите им дать. Рассмотрим пример файла карты:

```
# bogus networks
0.0.0.0/8          REJECT Mail server in broadcast network
10.0.0.0/8         REJECT No route to your RFC 1918 network
127.0.0.0/8        REJECT Mail server in Loopback network
224.0.0.0/4        REJECT Mail server in class D multicast network
192.168.0.0/16     REJECT No route to your RFC 1918 network
# spam havens
69.6.0.0/18        REJECT Listed on Register Of Known Spam Operations
# wildcard MTA
64.94.110.11/32   REJECT VeriSign Domain wildcard
```

Далее просто добавьте параметр *check\_sender\_mx\_access*, указав карту в качестве аргумента, в свой список *smtpd\_recipient\_restrictions*, например:

```
smtpd_recipient_restrictions =
reject_non_fqdn_recipient
reject_non_fqdn_sender
reject_unknown_sender_domain
reject_unknown_recipient_domain
```





```
permit_mynetworks
reject_unauth_destination
check_recipient_access hash:/etc/postfix/roleaccount_exceptions
check_helo_access pcre:/etc/postfix/helo_checks
check_sender_mx_access cidr:/etc/postfix/bogus_mx
permit
```

Ограничение вступит в силу после перезагрузки параметров. Его действие найдет отражение в почтовом журнале:

```
Sep 17 12:19:23 mail postfix/smtpd[3323]: A003D15C021: reject: RCPT from
unknown[61.238.134.162]:
554 <recipient@example.com>: Sender address rejected: VeriSign Domain
wildcard;
from=<alli.k_Lacey_mq@joymail.com> to=<recipient@example.com> proto=ESMTP
helo=<example.com>
```

Ограничение вступит в силу после перезагрузки параметров. Его действие найдет отражение в почтовом журнале.

### 21.3.11.3 Возврат множеству получателей

В разделе «Пустое имя отправителя конверта» вы узнали о том, что не следует блокировать сообщения с пустым именем отправителем конверта. Это правило имеет одно исключение – необходимо блокировать сообщения с пустым именем отправителя конверта, отправленные множеству получателей. Дело в том, что в настоящее время нет оснований для легальной рассылки уведомлений о состоянии многочисленным получателям, так что любые такие сообщения, вероятно, являются запрещенными.

Для отказа в приеме сообщениям с пустым именем отправителя конверта, предназначенным нескольким получателям, добавьте параметр `reject_multi_recipient_bounce` в свой список `smtpd_recipient_restrictions`. Этот параметр может быть добавлен практически в любое место списка ограничений; в данном примере он поставлен в `smtpd_data_restrictions`:

```
smtpd_data_restrictions =
    reject_multi_recipient_bounce
```

В документации говорится, что параметр `reject_multi_recipient_bounce` может надежно использоваться только в `smtpd_data_restrictions`, когда известны все





получатели. Проверить это ограничение вы, как и раньше, можете посредством ручного подключения к почтовому серверу. Передача пустого имени отправителя конверта и нескольких получателей приведет к отказу, как показано в следующем примере:

```
mail:~/Desktop # telnet mail.itstep.org 25
Trying 192.168.40.1...
Connected to mail.itstep.org.
Escape character is '^].
220 mail.itstep.org ESMTP Postfix
ehlo client
250-mail.itstep.org
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
mail from:<>
250 2.1.0 Ok
rcpt to:root@itstep.org
250 2.1.5 Ok
rcpt to:recipient@itstep.org
250 2.1.5 Ok
data
550 5.5.3 <DATA>: Data command rejected: Multi-recipient bounce
```

### 21.3.11.4 Использование черных списков DNS

DNS сервер черных списков – это сервер, который сообщает вам о ресурсах (IP адресах, отправителях конвертов и доменах), которым, вероятно, не стоит доверять. Правильно выбранные черные списки могут быть чрезвычайно полезны для блокирования почты, отправленной клиентами на ваш сервер. Однако неверный выбор черного списка может заставить ваш сервер отвергать почту, которую вы считаете допустимой. Обязательно проверяйте принципы составления черного списка, прежде чем им воспользоваться. Любой сайт, ведущий черный список, должен представлять перечень условий, по которым ресурс вносится в черный список; кроме того, должна быть опубликована процедура удаления из списка ресурса, который более не должен там находиться. Если вы ищете черный список, то можете начать с сайта dmoz.org: (<http://dmoz.org/Computers/Internet/E-mail/Spam/Blacklists>).

Следует понимать, что основой всех черных списков является служба доменных имен, т. е. Postfix должен выполнять поиск в DNS. Некэшированный поиск в DNS может





занять около секунды, и в случае тайм-аута скорость, с которой сервер может принимать сообщения, значительно уменьшается. Поэтому проверки по черным спискам достаточно затратны с точки зрения времени отклика. Вы должны использовать их в своем списке ограничений только в качестве последнего средства.

Вы можете отвергать занесенные в черный список клиенты при помощи списков DNSBL (DNS-based Blackhole List). В Postfix есть параметр *reject\_rbl\_client*, который принимает в качестве аргумента полное имя хоста сервера черных списков. Приведем пример использования этого параметра:

```
smtpd_recipient_restrictions =
reject_non_fqdn_recipient
reject_non_fqdn_sender
reject_unknown_sender_domain
reject_unknown_recipient_domain
permit_mynetworks
reject_unauth_destination
check_recipient_access hash:/etc/postfix/roleaccount_exceptions
check_helo_access pcre:/etc/postfix/helo_checks
reject_rbl_client relays.ordb.org
permit
```

Новый параметр вступит в силу после перезагрузки параметров. Для того чтобы проверить, упомянут ли клиент в списке DNSBL, измените на обратный порядок четырех октетов IP адреса клиента (т. е. замените a.b.c.d на d.c.b.a), добавьте в конец rbl.domain (например, *relays.ordb.org*) и ищите в списке полученное значение. Если хост занесен в черный список, то вы получите ответ, указывающий на исходный IP адрес, как в следующем примере:

```
mail:~/Desktop # host 2.0.0.127.relays.ordb.org
2.0.0.127.relays.ordb.org has address 127.0.0.2
```

В дополнение к запрещению почты от определенных IP адресов вы можете блокировать сообщения тех отправителей, домены которых занесены в черные списки. Такие списки называются RHSBL (Right-Hand-Side Blacklist – черный список правых частей). Настройка Postfix для использования RHSBL требует выполнения тех же операций, что и для DNSBL. Postfix имеет параметр *reject\_rhsbl\_sender*, который отделяет локальную часть электронного адреса и использует доменную часть для обращения к черному списку (такому как *dsn.rfc-ignorant.org*). Если домен отправителя конверта внесен в черный список, то Postfix отклоняет входящее сообщение. Как и





другие параметры черных списков, этот параметр должен быть помещен в конец списка ограничения *smtpd\_sender\_restrictions*, например:

```
smtpd_sender_restrictions =  
    reject_rhsbl_sender dsn.rfc-ignorant.org
```

После перезагрузки изменения вступят в силу, и вы сможете проверить ограничение, подключившись к вашему серверу и используя отправителя конверта из домена, входящего в список *rfc-ignorant.org*, как в следующем примере (*sender@example.com* – это официальный адрес для проверок):

```
mail:~/Desktop # telnet mail.itstep.org 25  
Trying 192.168.40.1...  
Connected to mail.itstep.org.  
Escape character is '^]'.  
220 mail.itstep.org ESMTP Postfix  
helo client  
250 mail.itstep.org  
mail from:sender@example.com  
250 2.1.0 Ok  
rcpt to:root@itstep.org  
554 5.7.1 Service unavailable; Sender address [sender@example.com] blocked  
using dsn.rfc-ignorant.org; Not supporting null originator (DSN)
```

Проверка домена на вхождение в RHSBL аналогична процедуре проверки IP адреса, с тем лишь отличием, что не нужно менять порядок элементов. Просто добавьте имя сервера черных списков после имени домена, который вы хотите проверить, и выполните поиск в DNS. Здесь проверка была проведена для домена, которого не оказалось в черном списке, а затем для домена, который в этом списке присутствует:

```
mail:~/Desktop # host linux.org.dsn.rfc-ignorant.org  
Host linux.org.dsn.rfc-ignorant.org not found: 3(NXDOMAIN)  
mail:~/Desktop # host example.com.dsn.rfc-ignorant.org  
example.com.dsn.rfc-ignorant.org has address 127.0.0.2
```

### 21.3.11.5 Проверка отправителя

Бриллиант в короне средств Postfix борьбы со спамом – это проверка адреса отправителя, в ходе которой проверяется, существует ли в домене отправителя адрес отправителя, и если такого отправителя не существует, то Postfix не принимает сообщение.



К сожалению, эта функциональность является весьма дорогостоящей, т. к. проверка занимает много времени и требует дополнительных системных ресурсов. Рассмотрим ее пошагово:

1. Клиент передает данные отправителя конверта.
  2. Postfix формирует и ставит в очередь пробное сообщение отправителю конверта.
  3. Postfix ищет MX или A-запись домена отправителя конверта.
  4. Postfix пытается подключиться к почтовому серверу отправителя.
- Если подключиться к удаленному серверу не удается, то smtpd откладывает решение о принятии сообщения, возвращая клиенту код временной ошибки 450. Тем временем Postfix продолжает пытаться проверить адрес.
5. Postfix инициирует сеанс SMTP с удаленным сервером.
  6. Postfix передает данные отправителя конверта удаленному почтовому серверу в качестве сведений о получателе конверта.
  7. В зависимости от ответа удаленного сервера Postfix делает одно из двух:
    - Если удаленный почтовый сервер принимает получателя (отправителя исходного конверта), то Postfix отключается, уничтожает пробное сообщение и принимает сообщение от исходного клиента.
    - Если удаленный почтовый сервер отклоняет получателя (отправителя исходного конверта), то Postfix отключается, уничтожает пробное сообщение и отклоняет сообщение от клиента.

При включенной проверке адресов отправителей сообщения обычно будут испытывать задержку до 9 секунд на проверку адреса, встретившегося впервые. Однако затем Postfix кэширует статус адреса, так что последующие сообщения не будут задерживаться. Если проверка длится более 9 секунд, то smtpd отвергает сообщение клиента (отправляющего компьютера) с кодом 450. Обычные почтовые клиенты повторят попытку через некоторое время, а захваченные (hijacked) прокси-серверы – нет, т. к. они занимаются только пересылкой команд SMTP, и человек, управляющий таким прокси-сервером, не захочет терять дополнительное время.

Для включения проверки адреса отправителя добавьте параметр `reject_unverified_sender` в свой список ограничений `smtpd_recipient_restrictions`, например:

```
smtpd_recipient_restrictions =  
reject_non_fqdn_recipient  
reject_non_fqdn_sender
```



```
reject_unknown_sender_domain
reject_unknown_recipient_domain
permit_mynetworks
reject_unauth_destination
reject_rbl_client relays.ordb.org
check_recipient_access hash:/etc/postfix/roleaccount_exceptions
check_helo_access pcre:/etc/postfix/helo_checks
check_sender_access hash:/etc/postfix/rhsbl_sender_exceptions
reject_rhsbl_sender dsn.rfc-ignorant.org
reject_unverified_sender
permit
```

Кроме *reject\_unverified\_sender* существуют и другие параметры, которые можно добавить в ограничения. Однако параметры обладают разумными значениями по умолчанию, и служат они скорее для регулировки проверки адреса отправителя, чем для ее настройки. Следующие подразделы описывают наиболее часто используемые изменения проверки адреса отправителя. Дополнительные параметры настройки вы сможете найти в файле ADDRESS\_VERIFICATION\_README, который присутствует в дистрибутиве Postfix.

Когда Postfix формирует пробное сообщение для проверки отправителя, он сам должен представиться удаленному серверу – указать своего отправителя конверта. Вы можете задать этот адрес в параметре *address\_verify\_sender*. Значение по умолчанию – *postmaster@\$myorigin*. При желании указать другого отправителя конверта для пробного сообщения добавьте параметр *address\_verify\_sender* в файл *main.cf*:

```
address_verify_sender = sender@example.com
```

Конечно, такой адрес отправителя должен существовать; не забывайте о том, что другие серверы также могут использовать по отношению к вам механизм проверки адреса отправителя.

По умолчанию Postfix хранит проверенные адреса отправителей в оперативной памяти. Когда вы перезагружаете параметры или перезапускаете Postfix, то теряете их, если только не будете использовать (что не обязательно) дополнительную базу данных для постоянного хранения адресов. Для того чтобы работать с базой данных, задайте путь к ней в параметре *address\_verify\_map* (убедившись, что в выбранной файловой системе достаточно много свободного места), например:

```
address_verify_map = btree:/var/spool/postfix/verified_senders
```





После перезагрузки Postfix создаст базу данных и будет добавлять туда результаты как положительных, так и отрицательных проверок. Если вы хотите отменить сбор отрицательных данных, задайте параметр *address\_verify\_negative\_cache* в файле *main.cf*:

```
address_verify_negative_cache = no
```

По мере возрастания нагрузки на ваш почтовый сервер процедура проверки адресов получателей, вероятнее всего, приведет к нехватке ресурсов. В этот момент следует перейти к выборочной проверке адресов отправителей.

Выборочная проверка адресов отправителей работает на основе карты обычно используемых спамерами доменов отправителей конвертов. Если домен отправителя входящего сообщения присутствует в карте, то Postfix проверяет отправителя, иначе – не беспокоится. Вам нужно создать файл карты, например */etc/postfix/common\_spam\_senderdomains*, и указать параметр *reject\_unverified\_sender* в качестве действия, которое следует предпринять в случае совпадения с доменом отправителя конверта. Приведем пример того, как может выглядеть такой файл:

```
hotmail.com reject_unverified_sender
web.de reject_unverified_sender
msn.com reject_unverified_sender
mail.ru reject_unverified_sender
```

Страница руководства *access(5)* поясняет, что правая часть карты – это имя действующего ограничения или класса *smtpd\_restriction\_class*.

В данном примере при инициировании клиентом передачи сообщения Postfix делает одно из двух:

- Если домену отправителя соответствует запись в *common\_spam\_sender\_domains*, то просмотр карты возвращает действие *reject\_unverified\_sender*, и Postfix проверяет отправителя конверта. В случае успешной проверки *reject\_unverified\_sender* возвращает DUNNO, и Postfix переходит к оценке следующего ограничения. Если же адрес не действителен, то Postfix отклоняет сообщение.
- Если домену отправителя не соответствует никакая запись в *common\_spam\_senderdomains*, то поиск в карте не дает результата, и выборочный





оценщик возвращает DUNNO, а Postfix оценивает следующее ограничение, не проверяя адрес отправителя.

После создания карты преобразуйте ее в базу данных, используя команду:

```
postmap hash:/etc/postfix/common_spam_senderdomains
```

Наконец, замените существующий параметр *reject\_unverified\_sender* на параметр *check\_sender\_access* с картой в качестве аргумента. Приведем пример, в котором используется карта *hash:/etc/postfix/common\_spam\_senderdomains*:

```
smtpd_recipient_restrictions =
reject_non_fqdn_recipient
reject_non_fqdn_sender
reject_unknown_sender_domain
reject_unknown_recipient_domain
permit_mynetworks
reject_unauth_destination
check_recipient_access hash:/etc/postfix/roleaccount_exceptions
check_helo_access pcre:/etc/postfix/helo_checks
check_sender_access hash:/etc/postfix/rhsbl_sender_exceptions
reject_rhsbl_sender dsn.rfc-ignorant.org
check_sender_access hash:/etc/postfix/common_spam_senderdomains
permit
```

Вы можете пойти дальше и ввести, помимо отправителя конверта, дополнительные критерии проверки, например содержимое сообщения. Создайте новую карту с именем *common\_spam\_senderdomain\_keywords* – она будет содержать ключевые слова из имен доменов, которые будут запускать проверку адреса отправителя, например:

```
/sex/ reject_unverified_sender
/girl/ reject_unverified_sender
/sell/ reject_unverified_sender
```

Затем добавьте еще один параметр *check\_sender\_access*, указывающий на новую карту:

```
smtpd_recipient_restrictions =
reject_non_fqdn_recipient
reject_non_fqdn_sender
reject_unknown_sender_domain
reject_unknown_recipient_domain
permit_mynetworks
```





```
reject_unauth_destination
check_recipient_access hash:/etc/postfix/roleaccount_exceptions
reject_non_fqdn_hostname
reject_invalid_hostname
check_helo_access pcre:/etc/postfix/helo_checks
check_sender_access hash:/etc/postfix/rhsbl_sender_exceptions
reject_rhsbl_sender dsn.rfcignorant.org
check_sender_access hash:/etc/postfix/common_spam_senderdomains
check_sender_access regexp:/etc/postfix/common_spam_senderdomain_keywords
permit
```

## 21.4 Использование встроенных фильтров содержимого

Postfix может исследовать содержимое сообщения на основе таблиц шаблонов и действий. Такие проверки предназначены только для простой фильтрации содержимого. Для решения более сложных задач следует использовать внешние фильтры, которые будут рассмотрены в следующем разделе.

Проверки ищут в сообщениях определенные символы, а также могут изменять сообщения. Имена параметров конфигурации, включающих проверки, заканчиваются на *\_checks*, и все они – *header\_checks*, *body\_checks*, *mime\_header\_checks* и *nested\_header\_checks* – действуют по одной схеме:

1. Postfix анализирует сообщение строку за строкой, сравнивая их с картой шаблонов, составленных из регулярных выражений (regexp) или регулярных выражений Perl (PCRE).
2. Если строка соответствует регулярному выражению, то Postfix предпринимает действие, определенное для данного выражения, и переходит к исследованию следующей строки ввода.

Чтобы проверить, какие карты поддерживает ваша версия Postfix, выполните команду *postconf -m* – будут выведены все типы карт, поддерживаемых системой. В приведенном примере Postfix поддерживает regexp и PCRE, а также ряд других карт:





```
mail:~/Desktop # postconf -m
btree
cidr
environ
hash
ldap
mysql
nis
pcre
proxy
regexp
sdbm
static
tcp
unix
```

Все системы должны по умолчанию поддерживать таблицы типа regexp. Если у вашей системы при использовании карт регулярных выражений возникают проблемы с производительностью (или, еще хуже, если реализация regexp в вашей системе содержит ошибки), то вы можете установить библиотеки и заголовочные файлы PCRE и пересобрать Postfix с поддержкой PCRE.

### 21.4.1 Безопасная реализация фильтрации

Регулярные выражения могут стать очень сложными, и может получиться так, что ваш шаблон не будет работать, ему будет соответствовать больше строк, чем предполагалось, или вы вообще перестанете понимать, что происходит.

Для отладки шаблонов Postfix предлагает действие WARN. Если вы используете это действие в правой части своего шаблона, то Postfix при совпадении с выражением будет доставлять сообщение и одновременно записывать комментарий в журнал электронной почты. После того как вы убедитесь, что шаблон работает, просто замените WARN на нужное действие. Безопасная процедура добавления проверок выглядит следующим образом:

1. Добавляете в карту свой шаблон с соответствующим ему действием WARN.
2. Создаете файл, содержащий выражение, совпадающее с шаблоном фильтра.
3. Проверяете, совпадает ли шаблон карты с тестовым шаблоном.
4. Включаете проверку в основном файле конфигурации Postfix.
5. Тестируете фильтр на реальных сообщениях.



## Добавление регулярного выражения и определение действия WARN

Первый этап – это добавление шаблона, который вы хотите проверять, в карту, и определение действия WARN на тот случай, когда содержимое сообщения будет совпадать с тестовым шаблоном. В этом разделе мы будем рассматривать пример, проверяющий на соответствие шаблону заголовок *Subject*, но вы сможете использовать эту процедуру и для других заголовков и других параметров *\*\_checks*. Добавьте шаблон фильтра в файл */etc/postfix/header\_checks*. Этот файл хранит карту для параметра *header\_checks*, например:

```
/^Subject: FWD: Look at pack from Microsoft/      WARN Unhelpful virus warning
```

### Создание тестового шаблона

Все, что вам нужно сделать для создания тестового шаблона, – это поместить совпадающее с шаблоном сообщение в файл, например */tmp/testpattern*. В нашем примере это будет такое сообщение:

```
From: dingdong@example.com
Subject: FWD: Look at pack from Microsoft
blah blah
```

### Соответствует ли регулярное выражение тестовому шаблону?

Проверим шаблон фильтра, передав карту проверки и тестовый шаблон команде *postmap*. Выполним, например, такую команду:

```
postmap -q - regexp:/etc/postfix/header_checks < /tmp/testpattern
```

В случае совпадения команда печатает соответствующую строку тестового шаблона, например:

```
Subject: FWD: Look at pack from Microsoft      WARN Unhelpful virus warning
```

Если соответствие шаблону не найдено, то команда *postmap* ничего не выводит.

### Определение проверки в основной конфигурации

Если пока все работает, вы можете отредактировать свой файл *main.cf*, использовав только что созданный и протестированный файл, содержащий *header\_checks*:

■ Киев	■ Днепропетровск	■ Львов	■ Ровно	■ Мариуполь	■ Полтава
■ Одесса	■ Донецк	■ Николаев	■ Запорожье	■ Луганск	■ Харьков





```
header_checks = regexp:/etc/postfix/header_checks
```

Перезагрузите конфигурацию и отправьте тестовое сообщение, содержащее тот же самый тестовый шаблон.

### Тестирование на реальном сообщении

Для проверки фильтра на реальном сообщении выполним команду для передачи нашего теста серверу Postfix:

```
/usr/sbin/sendmail recipient@example.com < /tmp/testpattern
```

Теперь посмотрим на почтовый журнал, чтобы проверить, записал ли Postfix сообщение для тестового шаблона. Одна из строк фрагмента журнала содержит предупреждение примерно следующего вида:

```
Mar 30 17:17:52 mail postfix/cleanup[2461]: 53CAB633B3: warning: header Subject: FWD: Look at pack from Microsoft from Local; from=<sender@example.com> to=<recipient@example.com>; Unhelpful virus warning
```

После того как вы убедитесь, что шаблон фильтра работает, вы можете без риска заменить действие WARN на то действие, которое на самом деле что-то делает, такое как REJECT или DISCARD.

## 21.4.2 Действия над сообщениями

### REJECT

Postfix может отклонять сообщения, используя действие REJECT. Вы можете применять это действие для блокировки сообщений, которые соответствуют шаблону, например, содержащих определенный заголовок Subject. Отклонение предотвращает попадание сообщений в систему, не подпуская их к требующему большого количества вычислений антивирусному средству, программе выявления спама или (что, возможно, еще хуже) к вашим пользователям.

### HOLD

- Киев
- Днепропетровск
- Львов
- Ровно
- Мариуполь
- Полтава
- Одесса
- Донецк
- Николаев
- Запорожье
- Луганск
- Харьков



Postfix может задержать доставку сообщений при помощи действия HOLD. Его можно использовать для помещения подозрительных сообщений «на удержание» для дальнейшей проверки. Для просмотра сообщений используйте команду `postcat`, а для отправки сообщения - команду `postsuper -H`. Для того чтобы удалить сообщение из очереди Postfix, используйте команду `postsuper -d`.

## IGNORE

Если вы хотите удалить строки из заголовков, используйте действие IGNORE. Вы можете использовать его для сокрытия сведений, записанных в заголовках (например, тип используемого вами почтового клиента) или для того, чтобы удалить заголовки Received, которые могли добавить ваши внутренние почтовые серверы, брандмауэры и антивирусные средства.

## DISCARD

Postfix может молча отбраковывать сообщения, используя действие DISCARD. Например, может понадобиться удалить сообщения с определенной строкой в поле темы так, чтобы никто этого не заметил. Удаляя сообщение, Postfix, как обычно, записывает действие в журнал.

## REDIRECT

Postfix может перенаправить сообщение другим получателям, используя действие REDIRECT в случае соответствия шаблону в заголовках и в теле сообщения.

Рассмотренные нами примеры и действия для анализа заголовков, аналогичным образом применяются и для анализа других частей электронного сообщения. Для проверки тела письма: `body_checks`, для проверки MIME-заголовков: `mime_header_checks`, для проверки заголовков вложений: `nested_header_checks`

## 21.5 Использование внешних фильтров содержимого

Описанные в предыдущих главах встроенные фильтры предназначены для решения простых проблем; более сложную фильтрацию поручают внешним программам. Postfix может запускать приложения проверки содержимого до или после постановки



сообщений в очередь. Если почта фильтруется до постановки в очередь, то Postfix может оставить уведомление отправителей на усмотрение клиента. Если же почта фильтруется после постановки в очередь, ответственность за уведомления несет Postfix.

Внешние фильтры содержимого вступают в дело, когда со сцены уходят встроенные фильтры заголовков и тела сообщения; внешние приложения не только исследуют и отвергают сообщения, но и могут изменять их содержимое. Стандартными задачами для внешних фильтров являются проверка сообщений на наличие вирусов и выявление спама.

Postfix поддерживает два механизма фильтрации: *content\_filter* – используется после постановки сообщения в очередь, и *smtpd\_proxy\_filter* – используется до постановки сообщения в очередь.

Оба подхода имеют свои недостатки:

- *content\_filter* генерирует дополнительный трафик, т. к. Postfix первоначально принимает сообщения перед обработкой. Это может впоследствии привести к необходимости возврата, если фильтрующее приложение решит, что сообщение должно быть отклонено.
- *smtpd\_proxy\_filter* сразу же отвергает нежелательное содержимое, но он плохо масштабируется и может оказаться недостаточно производительным.

Стандарты RFC говорят о том, что почтовый сервер должен решить, принять или отвергнуть сообщение, не позднее, чем на этапе команды DATA в диалоге SMTP. К сожалению, такое требование оставляет почтовому серверу мало времени на анализ содержимого сообщения, т. к. в почтовых клиентах реализован достаточно короткий тайм-аут с тем, чтобы не «зависнуть» в общении с неисправным почтовым сервером.

Например, тайм-аут для SMTP клиента Postfix определяется параметром *smtp\_data\_done\_timeout*, который весьма лоялен и по умолчанию установлен в 600 секунд. Если почтовый сервер завершает просмотр содержимого до истечения клиентского тайм-аута, все работает отлично, т. к. у сервера есть время на уведомление клиента о своем решении в отношении приема сообщения. Однако если сервер работает слишком медленно, то клиент заканчивает соединение и повторяет попытку позже, при этом шансы на успех при следующей попытке невелики.



Механизм *content\_filter* позволяет избежать таких проблем благодаря специальной организации исследования содержимого, таким образом, в системах современного предприятия предпочтительнее использовать именно данный механизм.

### 21.5.1 Amavisd-new

Amavisd-new - это программа, которая связывает почтовый агент с одним или несколькими средствами обнаружения вирусов или спама, такими как SpamAssassin, она активно развивается и рекомендована многочисленными администраторами почтовых систем. Для обеспечения функциональности обнаружения вирусов вам в дополнение к amavisd-new потребуется, по крайней мере, одна поддерживаемая антивирусная программа.

Для установки программы можно использовать стандартные средства, например:

```
zypper install amavisd-new
```

Для запуска и остановки *amavisd-new* следует использовать файл */etc/init.d/amavis*:

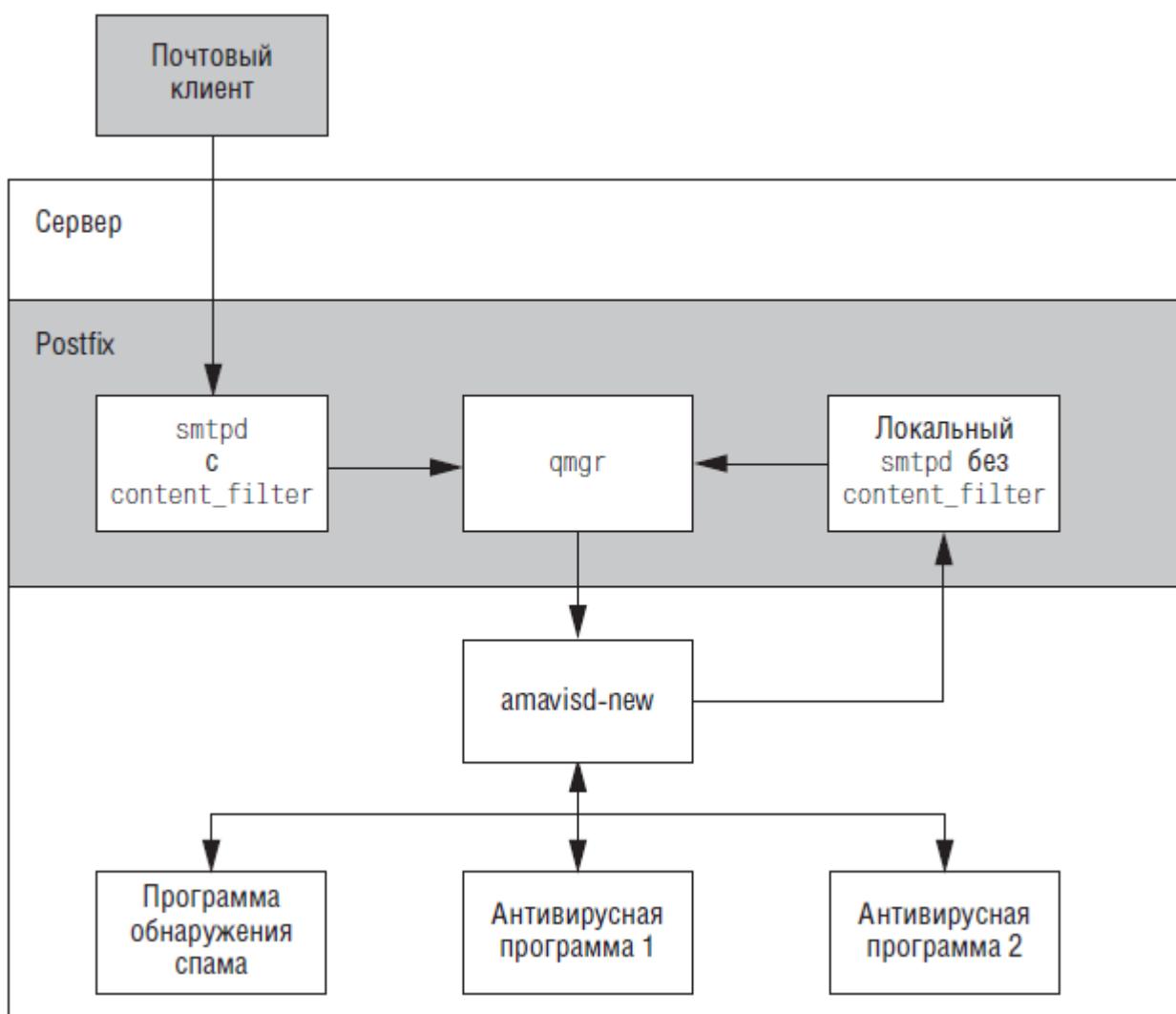
```
/etc/init.d/amavis {start/stop/status/reLoad}
```

Если вы используете *amavisd-new*, то сообщения на вашем сервере проделывают следующий путь:

1. Почтовый клиент отправляет сообщение *Postfix*.
2. *smtpd* принимает сообщение.
3. *smtpd* отправляет сообщение *qmgr*.
4. *qmgr* отправляет сообщение *amavisd-new*.
5. *amavisd-new* отправляет сообщение другим приложениям (например, антивирусным программам).
6. *amavisd-new* возвращает сообщение обратно локальному экземпляру *smtpd*.
7. Локальный *smtpd* отправляет сообщение *qmgr*.
8. *qmgr* или отправляет уведомление о возврате, или доставляет сообщение.

Данный путь хорошо проиллюстрирован на следующем изображении:





Прежде чем пытаться обеспечить взаимодействие *amavisd-new* с Postfix, следует протестировать работу данной программы автономно. Для этого необходимо выполнить следующие действия выполните сетевой тест, чтобы посмотреть, прослушивает ли она сетевой порт. Используйте сеансы telnet для тестирования как ESMT, так и LMTP взаимодействия.

## Проверка доступности ESMT

Открываем telnet соединение с локальным портом 10 024 (порт по умолчанию для *amavisd-new*). Вы должны убедиться, что программа прослушивает порт и отвечает на ESMT команды. В следующем примере сеанса программа *amavisd-new* отвечает на команду EHLO набором доступных команд:





```
mail:~/Desktop # telnet 127.0.0.1 10024
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^']'.
220 [127.0.0.1] ESMTP amavisd-new service ready
EHLO mail.itstep.org
250- [127.0.0.1]
250- VRFY
250-PIPELINING
250-SIZE
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
250 XFORWARD NAME ADDR PORT PROTO HELO SOURCE
quit
221 2.0.0 [127.0.0.1] amavisd-new closing transmission channel
Connection closed by foreign host.
```

## Проверка доступности LMTP

Затем следует проверить, прослушивает ли amavisd-new локальный порт 10024 (порт по умолчанию для LMTP) и отвечает ли на команды LMTP. Вы должны суметь запустить команду EHLO, как в данном сеансе:

```
mail:~/Desktop # telnet 127.0.0.1 10024
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^']'.
220 [127.0.0.1] ESMTP amavisd-new service ready
EHLO
250- [127.0.0.1]
250- VRFY
250-PIPELINING
250-SIZE
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
250 XFORWARD NAME ADDR PORT PROTO HELO SOURCE
quit
221 2.0.0 [127.0.0.1] amavisd-new closing transmission channel
Connection closed by foreign host.
```



## 21.5.2 Настройка Postfix для использования amavisd-new

На настоящий момент Postfix и amavisd-new работают независимо друг от друга. Следовательно, вам нужно настроить сервер Postfix так, чтобы он отправлял сообщения amavisd-new, и создать еще один экземпляр *smtpd* для возвращения сообщений в очередь Postfix. В дальнейших разделах будут описаны следующие этапы интегрирования amavisd-new в Postfix:

1. Создание транспорта.
2. Настройка транспорта.
3. Настройка пути возврата сообщений

Заметим, что отфильтрованная почта должна каким-то образом вернуться в очередь Postfix без повторного сканирования, поэтому вам необходим отдельный экземпляр *smtpd*, не использующий *content\_filter*. Это позволит amavisd-new возвращать сообщения в систему, не создавая бесконечных петель. Порт 25 уже занят, так что новый экземпляр *smtpd* должен прослушивать нестандартный порт. В нашем примере используется порт 10 025 локального хоста. Кроме того, amavisd-new также необходим порт для прослушивания. Тут вполне подходит значение по умолчанию – порт 10 024 локального хоста.

### Создание транспорта при помощи *content\_filter* в *main.cf*

Первый шаг в передаче обработки содержимого внешней программе – это определение транспорта, который отправляет сообщения фильтрующей программе. Postfix использует параметр *content\_filter* в файле *main.cf*. Параметр должен быть записан в виде *имя\_транспорта:следующий\_узел:порт*.

В рассматриваемом примере программа amavisd-new работает на том же компьютере, что и Postfix, так что вы можете обращаться к ней через порт 10024 локального хоста (127.0.0.1). Для того чтобы Postfix подключился к amavisd-new, необходимо определить в файле *main.cf* такой параметр *content\_filter*:

```
content_filter = amavisd-new:[127.0.0.1]:10024
```





## Определение транспорта в файле master.cf

Затем вам следует определить демон, который будет подключаться к *amavisd-new*, и создать его окружение. Это может быть демон *smtp*, *lmtp* или *pipe*. Если вы хотите использовать протокол *ESMTP* для отправки сообщений *amavisd-new*, то добавьте следующие записи в файл *master.cf*:

```
#=====
# service type      private unpriv chroot wakeup maxproc command
#           (yes)   (yes)   (yes)   (never) (100)
# =====
...
amavisd-new    unix      -      -      n      -      2      smtp
  -o smtp_data_done_timeout=1200s
  -o disable_dns_lookups=yes
```

Кое-что в приведенных записях необходимо отметить особо:

- Специальный транспорт *amavisd-new* – это копия обычного транспорта *smtp*. Его имя должно совпадать с именем транспорта, указанным для параметра *content\_filter*, который вы определили в файле *main.cf*.
- Программа *amavisd-new* требует достаточно большого объема ресурсов. Поэтому, если у вас не слишком быстрый компьютер, вы, возможно, захотите ограничиться максимум двумя одновременными экземплярами.
- Параметр *smtp\_data\_done\_timeout* является первой из двух дополнительных настроек, которые изменяют поведение демона. Обработка входящего сообщения может занять у *amavisd-new* значительное время, и увеличение тайм-аута после отправки демоном *smtp* сообщения не дает Postfix прекратить ожидание, пока *amavisd-new* не завершит работу.
- Вероятно, вы пока имеете дело лишь с локальными машинами, так что параметр *disable\_dns\_lookups* отменит ненужный поиск в DNS для клиента *smtp*.

Если вы решили использовать протокол *LMTP* (вместо *SMTP*) для передачи сообщений программе *amavisd-new*, то добавьте следующую запись в свой файл *master.cf*:





```
#=====
# service type      private  unpriv  chroot   wakeup   maxproc   command
#                   (yes)    (yes)    (yes)    (never) (100)
#=====

...
amavisd-new      unix      -        -        n        -        2          lmtp
-o lmtp_data_done_timeout=1200s
-o disable_dns_lookups=yes
```

Наконец, вам необходимо создать путь возврата, который позволит amavisd-new передавать сообщения обратно в очередь Postfix. Важно, что этот путь возврата игнорирует транспорт amavisd-new. В противном случае сообщение попало бы в замкнутый круг: Postfix отправлял бы сообщение amavisd-new, затем оно возвращалось бы в очередь Postfix, а оттуда снова отправлялось бы в amavisd-new. Путь возврата, игнорирующий любой определенный ранее параметр *content\_filter*, выглядит в файле *master.cf* следующим образом:

```
#=====
# service type      private  unpriv  chroot   wakeup   maxproc   command
#                   (yes)    (yes)    (yes)    (never) (100)
# =====

...
127.0.0.1:10025  inet      n        -        n        -        -          smtpd
-o content_filter=
-o local_recipient_maps=
-o relay_recipient_maps=
-o smtpd_restriction_classes=
-o smtpd_client_restrictions=
-o smtpd_helo_restrictions=
-o smtpd_sender_restrictions=
-o smtpd_recipient_restrictions=permit_mynetworks,reject
-o mynetworks=127.0.0.0/8
-o strict_rfc821_envelopes=yes
```

Из всех параметров предыдущей записи **абсолютно необходим** пустой параметр *content\_filter*. Эта настройка перекрывает параметр *content\_filter* в файле *main.cf*. Остальные параметры также подменяют соответствующие параметры *main.cf*, включая





параметры для отмены ограничений, которые не имеют смысла для транспорта, прослушивающего лишь локальный сетевой интерфейс.

Когда все настройки сделаны, можно приступить к тестированию фильтра. Не забывайте о том, что для вступления в силу изменений *master.cf* требуется перезагрузка конфигурации Postfix.

### 21.5.3 Тестирование фильтра amavisd-new в Postfix

Для тестирования совместной работы Postfix и amavisd-new вам следует проверить, может ли Postfix отправлять почту в amavisd-new и может ли amavisd-new возвращать сообщения обратно в очередь Postfix.

Тестирование будет состоять из следующих этапов:

1. Проверяем, прослушивает ли Postfix путь возврата.
2. Отправляем сообщение серверу Postfix и проверяем действия сервера: он должен направить сообщение в amavisd-new, а затем оно вернется обратно в очередь Postfix.
3. Проверяем, обнаружит ли антивирусный сканер тестовый образец.

#### Проверка пути возврата

Изменив файл *master.cf*, выполните команду *postfix reload*, чтобы Postfix прочитал измененный файл, а затем просмотрите файл журнала: нет ли там каких-нибудь жалоб. Затем проверьте, прослушивает ли демон возврата *smtpd* порт 10025 хоста *localhost*, как в следующем сеансе:

```
mail:~/Desktop # telnet 127.0.0.1 10025
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^].
220 mail.itstep.org ESMTP Postfix
ehlo client
250-mail.itstep.org
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
```



## Отправка тестового сообщения серверу Postfix

Postfix должен пропустить «неинффицированное» сообщение через всю систему. Отправляем сообщение из командной строки и следим за его судьбой по сообщениям в журнале, оставляемым там сервером Postfix и программой *amavisd-new*. Например, вы могли бы использовать такую команду для отправки своего файла *main.cf* по адресу *recipient@itstep.org*:

```
mail:~/Desktop # sendmail -f sender@itstep.org recipient@itstep.org < /etc/postfix/main.cf
mail:~/Desktop # tail /var/log/mail
Jan 17 17:23:07 mail postfix/smtpd[10886]: connect from localhost[127.0.0.1]
Jan 17 17:23:07 mail postfix/smtpd[10886]: C9A37F1CC: client=localhost[127.0.0.1]
Jan 17 17:23:07 mail postfix/cleanup[10872]: C9A37F1CC: message-id=<20110117152246.A7B02F1CF@mail.itstep.org>
Jan 17 17:23:07 mail postfix/qmgr[10225]: C9A37F1CC: from=<sender@itstep.org>, size=29732, nrcpt=1 (queue active)
Jan 17 17:23:07 mail postfix/smtpd[10886]: disconnect from localhost[127.0.0.1]
Jan 17 17:23:07 mail amavis[10270]: (10270-03) Passed CLEAN, <sender@itstep.org> -> <recipient@itstep.org>, Message-ID: <20110117152246.A7B02F1CF@mail.itstep.org>, mail_id: Q7F6Kptsxygo, Hits: 2 .499, size: 29083, queued_as: C9A37F1CC, 21055 ms
Jan 17 17:23:07 mail postfix/lmtp[10880]: A7B02F1CF: to=<recipient@itstep.org>, relay=127.0.0.1[127.0.0.1]:10024, delay=21, delays=0.22/0.06/0.02/21, dsn=2.0.0, status=sent (250 2.0.0 Ok, id=10270-03, from MTA([127.0.0.1]:10025): 250 2.0.0 Ok: queued as C9A37F1CC)
Jan 17 17:23:07 mail postfix/qmgr[10225]: A7B02F1CF: removed
Jan 17 17:23:07 mail postfix/local[10898]: C9A37F1CC: to=<recipient@itstep.org>, relay=local, delay=0.18, delays=0.07/0.1/0/0.01, dsn=2.0.0, status=sent (delivered to mailbox)
Jan 17 17:23:07 mail postfix/qmgr[10225]: C9A37F1CC: removed ¶
```

Обратите внимание на сообщение от *amavis*: “Passes CLEAN”. В сообщении не было обнаружено вирусов, и оно было передано дальше.

## Проверка тестового вирусного образца

Последней проверкой будет имитация заражения сообщения вирусом. Вы можете сделать это, получив тестовый образец вируса EICAR (<http://www.eicar.org>) и отправив его Postfix. Любые средства поиска вирусов, в которых этот образец специально не отключен, должны его распознавать. Например, следующая команда отправит вирус по адресу *recipient@example.com*:

```
mail:~/Desktop # sendmail -f sender@itstep.org recipient@itstep.org < /tmp/eicar.com
```

В результате, *amavis* реагирует на вирус, блокирует письмо и посыпает сообщение об этом событии на адрес *virusalert@itstep.org*:

- |          |                  |            |             |             |           |
|----------|------------------|------------|-------------|-------------|-----------|
| ■ Киев   | ■ Днепропетровск | ■ Львов    | ■ Ровно     | ■ Мариуполь | ■ Полтава |
| ■ Одесса | ■ Донецк         | ■ Николаев | ■ Запорожье | ■ Луганск   | ■ Харьков |





```
mail:~/Desktop # tail /var/log/mail
Jan 17 17:27:31 mail postfix/smtpd[10965]: connect from localhost[127.0.0.1]
Jan 17 17:27:31 mail postfix/smtpd[10965]: A205CF1CC: client=localhost[127.0.0.1]
Jan 17 17:27:31 mail postfix/cleanup[10952]: A205CF1CC: message-id=<VA0svCRDKjp2ZS@mail.itstep.org>
Jan 17 17:27:31 mail postfix/qmgr[10225]: A205CF1CC: from=<virusalert@itstep.org>, size=2898, nrcpt=1 (queue active)
Jan 17 17:27:31 mail postfix/smtpd[10965]: disconnect from localhost[127.0.0.1]
Jan 17 17:27:31 mail amavis[10269]: (10269-03) Blocked INFECTED (Eicar-Test-Signature), <sender@itstep.org> -> <recipient@itstep.org>, quarantine: virus-0svCRDKjp2ZS, Message-ID: <20110117152731.211BAF1CF@mail.itstep.org>, mail_id: 0svCRDKjp2ZS, Hits: -, size: 344, 442 ms
Jan 17 17:27:31 mail postfix/lmtp[10960]: 211BAF1CF: to=<recipient@itstep.org>, relay=127.0.0.1[127.0.0.1]:10024, delay=0.73, delays=0.22/0.05/0.02/0.44, dsn=2.7.0, status=sent (250 2.7.0 Ok, discarded, id=10269-03 - VIRUS: Eicar-Test-Signature)
Jan 17 17:27:31 mail postfix/qmgr[10225]: 211BAF1CF: removed
Jan 17 17:27:31 mail postfix/local[10973]: A205CF1CC: to=<admin@itstep.org>, orig_to=<virusalert@itstep.org>, relay=local, delay=0.13, delays=0.03/0.09/0/0.01, dsn=2.0.0, status=sent (delivered to mailbox)
Jan 17 17:27:31 mail postfix/qmgr[10225]: A205CF1CC: removed
```

## 21.6 SMTP аутентификация

SMTP аутентификация – это способ идентификации клиентов независимо от их IP адресов; он позволяет серверу пересыпалть сообщения от почтовых клиентов, чьи IP адреса не входят в список доверенных. Для поддержки SMTP аутентификации в Postfix необходимо использовать пакет Cyrus SASL.

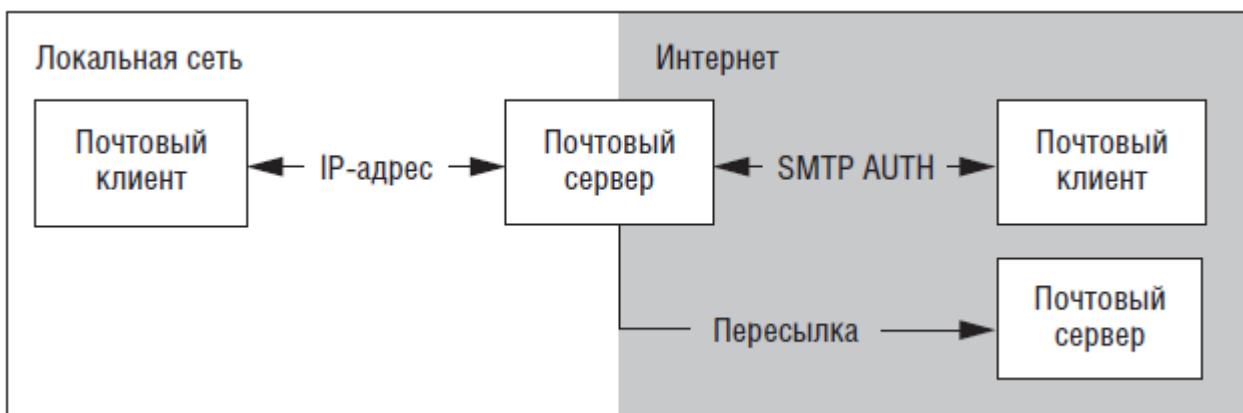
Первые серверы SMTP пересыпали почту от любого клиента любому получателю. С возникновением проблемы спама агенты передачи сообщений обрели возможность принимать запросы на ретрансляцию только от определенных клиентов. Разработчики агентов передачи сообщений приняли решение идентифицировать такие клиенты по их IP адресам, а администраторам оставалось только настроить свои системы так, чтобы они отвергали клиенты, не попавшие в список доверенных.

Сегодня попытки злоупотребления почтовыми рассылками остаются насущной проблемой, заставляющей администраторов тратить немало времени на усиление защиты своих серверов и введение дополнительных ограничений. К тому же привязка разрешений на пересылку к IP адресу становится все более трудоемкой по мере роста размеров и усложнения структуры современных сетей, кроме того, она совершенно не подходит для мобильных пользователей. Мобильные пользователи (в соответствии с определением в RFC 2977) нуждаются в доступе к ресурсам своего домена из любой точки Интернета. К сожалению, такие пользователи почти никогда не используют один и тот же IP адрес, более того, ни они сами, ни администратор почтового сервера не



знают заранее, какими будут их адреса, что делает бессмысленным применение правил, основанных на статических IP адресах.

SMTP аутентификация решает проблему «на корню»:



Основные этапы работы метода SMTP AUTH:

1. SMTP сервер предлагает почтовому клиенту пройти аутентификацию SMTP AUTH.
2. Клиент передает свои верительные данные серверу.
3. Сервер проверяет данные клиента и, если они верны, разрешает пересылку.

### 21.6.1 Cyrus SASL

Postfix реализует SMTP аутентификацию при помощи протокола SASL (Simple Authentication and Security Layer – простой протокол аутентификации и безопасности). Существует несколько реализаций SASL; Postfix использует библиотеки Cyrus SASL, которые происходят от первоначальной реализации SASL в проекте Cyrus. Cyrus SASL предоставляет множество разнообразных механизмов SASL, которые отличаются способом передачи верительных данных и уровнем обеспечиваемой безопасности. Практика показывает, что в среде, где необходимо поддерживать клиенты операционных систем Windows, Mac OS и Linux, лучше всего использовать механизмы PLAIN, LOGIN и CRAM-MD5.

Верительные данные, отправленные с использованием механизма PLAIN, передаются открытым текстом в кодировке base64. Это простой механизм, который реализует большинство почтовых клиентов, но строки в кодировке base64 легко поддаются декодированию. Механизм LOGIN – это тот же PLAIN, только он



используется для почтовых клиентов, которые не соответствуют RFC (как Outlook Express). В основе CRAM-MD5 лежит то, что клиент и сервер объединяют общий секретный ключ, обычно это пароль. Сервер формирует запрос на основе секретных данных, а клиент отвечает, доказывая, что секретный ключ ему известен. Это гораздо более безопасно, чем просто отправлять незашифрованный пароль по сети, но серверу все еще необходимо хранить этот секрет.

### 21.6.1.1 Методы аутентификации

Методы аутентификации (называемые также службами проверки паролей) управляют механизмами, заботясь о взаимодействии между приложением, которое обеспечивает SMTP аутентификацию, и хранилищем верительных данных.

Cyrus SASL предоставляет две службы проверки паролей: saslauthd и auxprop. Эти службы отличаются поддерживаемыми механизмами и хранилищами, к которым они могут подключаться. Приложение, предоставляющее аутентификацию через свой интерфейс, должно выбрать, какой службой проверки паролей воспользоваться

*saslauthd* – это автономный демон, который может быть запущен с привилегиями суперпользователя. Он может подключаться к различным видам хранилищ, но главным образом к тем, которые требуют привилегий суперпользователя (например, /etc/shadow). Использование saslauthd ограничивается механизмами открытого текста (PLAIN и LOGIN).

*auxprop* – это общая служба проверки паролей для ряда вспомогательных плагинов. Каждый плагин специализируется на определенном типе хранилища аутентификационных данных, такого как серверы SQL и sasldb2.

### 21.6.1.2 Хранилища аутентификационных данных

Cyrus SASL для проверки верительных данных, полученных от клиента, требует наличия одного или нескольких хранилищ аутентификационных данных. Служба проверки паролей проверяет, соответствуют ли верительные данные клиента тем, что находятся в хранилище. Перечислим хранилища аутентификационных данных, которые включены в официальный список поддерживаемых Cyrus SASL:

- Киев   ■ Днепропетровск   ■ Львов   ■ Ровно   ■ Мариуполь   ■ Полтава
- Одесса   ■ Донецк   ■ Николаев   ■ Запорожье   ■ Луганск   ■ Харьков



### **imap**

Сервер IMAP может проверять верительные данные.

### **kerberos**

Cyrus SASL может проверять мандаты Kerberos.

### **ldap**

Cyrus SASL может читать верительные данные с сервера OpenLDAP.

### **pam**

Cyrus SASL может читать данные из любых модулей, доступных ему посредством PAM (Pluggable Authentication Modules – подключаемые модули аутентификации).

### **passwd/shadow**

Cyrus SASL может читать данные из баз данных системного пользователя (/etc/passwd и, возможно, /etc/shadow).

### **sasldb2**

Cyrus SASL имеет собственную базу данных с именем sasldb2. Эта база данных необходима для сервера Cyrus IMAP, но он вам не нужен при использовании базы данных для SMTP аутентификации.

### **sql**

Cyrus SASL может обращаться к пользовательским данным на серверах SQL. В настоящее время поддерживаются серверы MySQL и PostgreSQL.

## **21.6.1.3      Настройка Cyrus SASL**

Для настройки Cyrus SASL необходимо проделать следующие действия:

### **Установите Cyrus SASL.**

Для этого установите пакеты под названием *cyrus-sasl-saslauthd* и *cyrus-sasl*.

### **Создайте конфигурационный файл приложения Postfix.**

Каждому приложению, предоставляющему услуги SASL, необходимо сообщить, как использовать библиотеки SASL. Cyrus SASL вместо одного большого общего файла конфигурации имеет по одному файлу для каждого приложения. Это позволяет



определить различные конфигурации для разных приложений. Файл конфигурации приложения для Postfix называется `smtpd.conf`, т. к. по умолчанию приложением Postfix, предоставляющим услуги SASL, является демон `smtpd`. В разных дистрибутивах может быть различным местоположение данного файла конфигурации. В некоторых операционных системах файл `smtpd.conf` содержит ряд настроек по умолчанию, так что заранее проверьте его наличие. Если файл не существует, создайте его от имени `root`. В Suse Linux Enterprise Server адрес данного файла следующий: `/etc/sasl2/smtpd.conf`.

## Установите уровень журналирования.

Первый параметр, который нужно указать в файле `smtpd.conf`, – это уровень журналирования (параметр `log_level`). Возможные значения параметра:

Значение <code>log_level</code>	Описание
0	Не вести журналирование
1	Записывать необычные ошибки (значение по умолчанию)
2	Записывать все неудачные попытки аутентификации
3	Записывать предупреждения о не фатальных ошибках
4	Более подробно, чем 3
5	Более подробно, чем 4
6	Записывать трассировку внутренних протоколов
7	Записывать трассировку внутренних протоколов, включая пароли

Когда вы настроите и протестируете Cyrus SASL, нужно будет установить уровень журналирования в файле `smtpd.conf` не менее 3:

`Log_Level: 3`

## Определите службу проверки паролей.

Теперь нужно сообщить серверу Postfix, какая служба проверки паролей будет использоваться для аутентификации пользователей. Вы должны точно определиться с тем, `saslauthd` или `auxprop` вы будете использовать, т. к. дальнейшие действия зависят от выбранной службы. В Cyrus SASL служба проверки паролей определяется параметром `pwcheck_method`. Если вы планируете использовать `saslauthd`, добавьте в `smtpd.conf` такую строку:





```
Log_Level: 3
pwcheck_method: saslauthd
```

Если же вы хотите использовать вспомогательный плагин, ваш файл smtpd.conf будет таким:

```
Log_Level: 3
pwcheck_method: auxprop
```

### Выберите механизмы SMTP AUTH.

Cyrus SASL позволяет клиенту выбрать механизмы, которые будут использоваться для аутентификации. В определенных обстоятельствах это может привести к ошибкам аутентификации:

- Если вы предлагаете механизмы, требующие настройки, которая у вас не выполнена. Например, если вы не используете Kerberos, но ваш сервер предлагает этот механизм и клиент выбирает его, то аутентификация не будет успешной.
- Если вы выберете в качестве службы проверки паролей saslauthd, но не ограничите предлагаемые механизмы механизмами открытого текста. В данном случае аутентификация не будет успешной, если клиент выберет другой механизм, т. к. saslauthd может работать только с механизмами PLAIN и LOGIN.

Параметр *mech\_list* позволяет вам гарантировать, что ваш сервер предлагает определенный список механизмов. Например, если вы используете *saslauthd*, то файл smtpd.conf должен быть таким:

```
Log_Level: 3
pwcheck_method: saslauthd
mech_list: PLAIN LOGIN
```

Для вспомогательных плагинов следует выбрать другой список, например так:

```
Log_Level: 3
pwcheck_method: auxprop
mech_list: PLAIN LOGIN CRAM-MD5 DIGEST-MD5
```

По завершению всех настроек, запустите службу *saslauthd*.



## 21.6.2 Настройка Postfix

В этом разделе будет рассказано о том, как настроить Postfix сервер *smtpd* для предоставления SMTP AUTH почтовым клиентам. После успешной аутентификации клиенты смогут пересыпаль сообщения через сервер Postfix, даже если их IP адреса не относятся к диапазону, определенному в параметре конфигурации *inetnetworks*.

Для настройки взаимодействия Postfix и Cyrus SASL необходимо проделать следующие действия:

### **Включить серверную часть SMTP AUTH.**

Первое, что необходимо сделать, – это включить SMTP аутентификацию для Postfix сервера *smtpd*, т. к. по умолчанию данная функциональность отключена. Используем параметр *smtpd\_sasl\_auth\_enable* в файле */etc/postfix/main.cf*:

```
smtpd_sasl_auth_enable = yes
```

### **Настройте механизмы SASL, которые будут предоставляться клиентам.**

Теперь следует определить механизмы аутентификации, которые сервер Postfix будет предлагать почтовым клиентам. Cyrus SASL предоставляет множество механизмов, начиная с анонимной «аутентификации» и заканчивая очень мощными системами, такими как Kerberos.

Управление предоставляемыми механизмами осуществляется параметром *smtpd\_sasl\_security\_options*. Вы можете указать в нем разделенный запятыми список из одного или более значений, описанных далее:

#### **noanonymous**

Это значение гарантирует, что сервер действительно проверяет верительные данные клиента. Это настройка по умолчанию, которую необходимо сохранить, т. к. некоторым спамерам известно об анонимной SMTP аутентификации. Список значений вашего параметра *smtpd\_sasl\_security\_options* всегда должен включать в себя *noanonymous*; в противном случае ваш почтовый сервер почти наверняка превратится в открытый ретранслятор.





### **noplaintext**

Добавлением в список значения *noplaintext* вы исключаете использование всех механизмов открытого текста, таких как PLAIN и LOGIN. Рекомендуется его использовать, т. к. отправляемые открытым текстом верительные данные могут быть легко перехвачены в сети.

### **noactive**

Это значение исключает использование механизмов SASL, которые восприимчивы к активным (не по словарю) атакам. Например, взаимная аутентификация не чувствительна к активным атакам.

### **nodictionary**

Исключаются все механизмы, не устойчивые к атакам по словарю. Атакующий по словарю использует грубую силу, пробуя пароль за паролем, пока один из них не подойдет.

### **mutual\_auth**

Использование *mutual\_auth* означает поддержку только механизмов, обеспечивающих взаимную аутентификацию. В этом случае и сервер должен аутентифицировать себя для клиента. При тестировании конфигурации изменять данный параметр не следует; значение по умолчанию *smtpd\_sasl\_security\_options = noanonymous* охраняет вас от спамеров, но разрешает применение механизмов открытого текста, что несколько облегчает отладку. Затем, когда вы убедитесь, что SMTP AUTH работает, следует отключить механизмы открытого текста, дополнив список значений параметра *smtpd\_sasl\_security\_options* опцией *noplaintext*.

*smtpd\_sasl\_security\_options = noanonymous*

## **Настройте поддержку SMTP AUTH для нестандартных почтовых клиентов.**

Следующее, что, вероятно, стоит сделать, – это настроить поддержку альтернативной нотации в SMTPдиалоге для «устаревших» почтовых клиентов, которые, однако, могут использовать SMTP AUTH. «Устаревшие» почтовые клиенты не распознают SMTP AUTH, когда она предлагается так, как указано в RFC 2222, но они распознают более раннюю нотацию, использованную в черновом варианте этого стандарта, где между командой AUTH и названиями механизмов стоял не пробел, а знак равенства (=).





Для поддержки таких почтовых клиентов установите параметр *broken\_sasl\_auth\_clients* в файле *main.cf*:

```
broken_sasl_auth_clients = yes
```

#### Определите разрешения на пересылку в Postfix.

Выполняем последнюю операцию – разрешаем пересылку для клиентов, прошедших аутентификацию SASL. Для этого добавляем параметр *permit\_sasl\_authenticated* в список ограничений *smtpd\_recipient\_restrictions* своей конфигурации, например, так:

```
smtpd_recipient_restrictions =
[...]
permit_sasl_authenticated,
permit_mynetworks,
reject_unauth_destination
[...]
```

Необходимо поместить ключевое слово *permit\_sasl\_authenticated* достаточно близко к началу списка ограничений, чтобы аутентифицированный клиент не был случайно отвергнут из-за несоответствия какому-то другому правилу (главным образом речь идет о *reject\_unauth\_destination*). Базовая настройка SMTP AUTH на сервере завершена. Перезагружаем конфигурацию и приступаем к тестированию.

### 21.6.3 Тестирование конфигурации

Тестирование SMTP аутентификации на стороне сервера включает в себя следующие этапы:

1. Проверить SMTP диалог, чтобы убедиться в том, что *smtpd* предоставляет SMTP AUTH.
2. Аутентифицировать пользователя, чтобы убедиться в том, что сервер Postfix может взаимодействовать с Cyrus SASL.
3. Отправить тестовое сообщение удаленному пользователю, чтобы проверить, могут ли аутентифицированные пользователи пересыпать сообщения нелокальным пользователям через наш сервер.

#### Проверка SMTP-диалога

■ Киев	■ Днепропетровск	■ Львов	■ Ровно	■ Мариуполь	■ Полтава
■ Одесса	■ Донецк	■ Николаев	■ Запорожье	■ Луганск	■ Харьков





Убедимся в том, что сервер Postfix предоставляет SMTP AUTH почтовым клиентам, так что клиенты знают, когда они могут инициировать SMTP аутентификацию. Подключитесь к своему серверу и отправьте ему приветствие EHLO (SMTP AUTH работает только в расширенном протоколе SMTP), например:

```
mail:~/Desktop # telnet mail.itstep.org 25
Trying 192.168.40.1...
Connected to mail.itstep.org.
Escape character is '^].
220 mail.itstep.org ESMTP Postfix
ehlo client.itstep.org
250-mail.itstep.org
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-AUTH PLAIN LOGIN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
```

Как видите, параметр AUTH не только сообщает вам о включении SMTP AUTH, но также предлагает перечень поддерживаемых механизмов аутентификации.

## Аутентификация пользователя

Для аутентификации пользователя вам понадобится строка в кодировке base64, содержащая действительные имя и пароль из вашего хранилища аутентификационных данных. Например, для пользователя с именем *test* и паролем *testpass* используйте такую команду:

```
perl -MMIME::Base64 -e 'print encode_base64("test\0test\0testpass");'
```

Результатом ее выполнения будут такие данные:

```
mail:~/Desktop # perl -MMIME::Base64 -e 'print encode_base64("test\0test\0testpass");'
dGVzdABOZXNOAHRlc3RwYXNz
```

Теперь подключаемся к серверу и с помощью команды EHLO устанавливаем расширенное SMTP соединение, затем используем AUTH PLAIN, чтобы сообщить серверу Postfix о том, что мы хотим аутентифицироваться при помощи механизма

- |          |                  |            |             |             |           |
|----------|------------------|------------|-------------|-------------|-----------|
| ■ Киев   | ■ Днепропетровск | ■ Львов    | ■ Ровно     | ■ Мариуполь | ■ Полтава |
| ■ Одесса | ■ Донецк         | ■ Николаев | ■ Запорожье | ■ Луганск   | ■ Харьков |





открытого текста и передать строку в кодировке base64. Приведем пример успешного теста:

```
mail:~/Desktop # telnet mail.itstep.org 25
Trying 192.168.40.1...
Connected to mail.itstep.org.
Escape character is '^]'.
220 mail.itstep.org ESMTP Postfix
ehlo client.itstep.org
250-mail.itstep.org
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-AUTH PLAIN LOGIN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
auth plain dGVzdAB0ZXNOAHRlc3RwYXNz
235 2.7.0 Authentication successful
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

Мы видим подтверждение успешной аутентификации – строку 235 Authentication successful.

## Пересылка тестового сообщения

Наконец, нужно проверить, что сервер Postfix разрешает аутентифицированному пользователю пересылать сообщения. Однако для начала следует убедиться в том, что другие разрешения на пересылку не мешают вашим новым правилам, связанным с аутентификацией. Для этого подключайтесь к серверу с хоста или из сети, которым не разрешена пересылка без SMTP аутентификации. Имеет смысл перепроверить все дважды, так что сначала попробуйтесь отправить сообщение без использования SMTP AUTH.

Если у вас нет доступа к клиенту вне сетей, определенных в параметре *mynetworks*, то отключите параметр *mynetworks* и задайте на время тестирования *mynetworks\_style = host*. В этом случае разрешения на пересылку будут ограничены лишь сервером, так что вы сможете использовать для тестирования сервера любой хост своей локальной сети.



Подключаемся к серверу, как и в предыдущем разделе, но не завершаем соединение после успешной аутентификации, а отправляем сообщение нелокальному пользователю.

Если сервер отправляет в качестве ответа команде RCPT TO: сообщение 250 Ok, и вы с помощью команды DATA смогли отправить сообщение, значит, аутентификация работает корректно.

## 21.7 POP3/IMAP сервер Dovecot

При создании Dovecot, авторы, прежде всего, держали в уме его безопасность. Это действительно безопасный и, кроме того, быстрый, простой в настройке POP3/IMAP сервер, который использует очень мало оперативной памяти.

POP3/IMAP сервер необходим для того, чтобы клиенты могли получить доступ к своим почтовым ящикам, которые хранятся на сервере под управлением Postfix. Для этого пользователи должны пройти процедуру аутентификации.

Для установки Dovecot достаточно установить одноименный пакет. В имени пакета может присутствовать номер его версии:

```
mail:~/Desktop # zypper se -d *dovecot*
Загрузка данных репозитария...
Чтение установленных пакетов...

S | Имя                                | Заключение          | Тип
--+
i | dovecot12                           | IMAP and POP3 Server Written Primarily with Security in -> | пакет
i | dovecot12-backend-mysql             | MySQL support for Dovecot                         | пакет
i | dovecot12-backend-pgsql            | PostgreSQL support for Dovecot                      | пакет
i | dovecot12-backend-sqlite           | SQLite support for Dovecot                        | пакет
| dovecot12-devel                     | Development files for Dovecot plugins           | пакет
| dovecot12-fts-lucene               | Fulltext search support via CLucene              | пакет
| dovecot12-fts-solr                 | Fulltext search support via solr                  | пакет
```

### 21.7.1 /etc/dovecot/dovecot.conf

Рассмотрим самую простую конфигурацию аутентификации, которая будет использоваться на нашем почтовом сервере: PLAIN. Для этого в конфигурационном файле */etc/dovecot/dovecot.conf* указываем следующую команду:

```
disable_plaintext_auth = no
```





Так как выбранный тип аутентификации не подразумевает специального шифрования, отключим поддержку *ssl*:

```
ssl = no
```

Укажем названия протоколов, которые должен использовать сервер:

```
protocols = imap pop3
```

Для тестирования укажем особый уровень логирования сообщений, после завершения этапа тестирования его можно будет отключить:

```
mail_debug = yes
```

Настроим Dovecot и Postfix таким образом, чтобы почтовые сообщения хранились в домашних каталогах локальных пользователей системы. Для этого в *dovecot.conf* указываем следующий параметр:

```
mail_location = maildir:~/Maildir
```

А в *main.cf* - конфигурационном файле Postfix – зададим необходимое значение параметру *home\_mailbox*:

```
home_mailbox = Maildir/
```

Как уже было сказано, для аутентификации пользователей мы будем использовать механизм PLAIN, при этом пользователи должны будут предоставлять учетные данные, которые соответствуют локальным учетным записям сервера. Для этого задаем следующий ряд конфигурационных параметров файла *dovecot.conf*:

```
auth default {
mechanisms = plain      # используемый по умолчанию механизм аутентификации - PLAIN

passdb pam {            # Хранилище аутентификации - pam
}
userdb passwd {          # Используются локальные системные пользователи
}
user = root              # Доступ к /etc/passwd осуществляется от имени root, что
# необходимо при использовании хранилища pam
}
```



## 21.7.2 Тестирование конфигурации

После того, как все настроено, следует перезагрузить настройки серверов Dovecot и Postfix и приступить к этапу проверки настроек. Для тестирования конфигурации, прежде всего, отправьте письмо локальному пользователю вашего сервера, например пользователю *test*. Затем следует обратиться на POP3 порт нашего сервера:

```
mail:~/Desktop # telnet mail.itstep.org 110
Trying 192.168.40.1...
Connected to mail.itstep.org.
Escape character is '^].
+OK Dovecot ready.
user test
+OK
pass testpass
+OK Logged in.
retr 1
+OK 1132 octets
Return-Path: <root@itstep.org>
X-Original-To: test@itstep.org
Delivered-To: test@itstep.org
Received: from localhost (localhost [127.0.0.1])
    by mail.itstep.org (Postfix) with ESMTP id 12CDFF2D7
        for <test@itstep.org>; Wed, 19 Jan 2011 13:46:37 +0200 (EET)
X-Virus-Scanned: amavisd-new at itstep.org
X-Spam-Flag: NO
X-Spam-Score: 2.207
X-Spam-Level: **
X-Spam-Status: No, score=2.207 tagged_above=2 required=6.2 tests=[AWL=-0.289,
    LOCALPART_IN SUBJECT=2.497, NO_RELAYS=-0.001] autolearn=no
Received: from mail.itstep.org ([127.0.0.1])
    by localhost (mail.itstep.org [127.0.0.1]) (amavisd-new, port 10024)
        with LMTP id C7IFEEaSB-W93 for <test@itstep.org>;
        Wed, 19 Jan 2011 13:46:27 +0200 (EET)
Received: by mail.itstep.org (Postfix, from userid 0)
    id B08ADF2D9; Wed, 19 Jan 2011 13:46:27 +0200 (EET)
Date: Wed, 19 Jan 2011 13:46:27 +0200
To: test@itstep.org
```

Используя синтаксис протокола POP3, проходим процесс аутентификации (команды *user admin* и *pass testpass*), а также просматриваем первое письмо (*retr 1*), которое хранится в почтовом ящике. На изображении представлен процесс успешной аутентификации и просмотр сообщения.





## 21.8 SquirrelMail

Наша почтовая система работает, осталось лишь слегка усовершенствовать ее, добавив пользователям возможность работы с помощью Web-интерфейса. SquirrelMail - это система доступа пользователей к возможностям SMTP и IMAP сервера с помощью Web интерфейса. Для ее получения и установки необходимо выполнить следующие действия:

1. Скачайте актуальную версию squirrelmail с сайта производителя:  
<http://squirrelmail.org/>
2. Распакуйте содержимое архива в каталог /srv/www/htdocs/squirrelmail
3. Перейдите в каталог /srv/www/htdocs/squirrelmail и выполните команду:  
`.configure`
4. В появившемся меню укажите все необходимые настройки, например DNS имя нашего почтового сервера. Сохраните конфигурацию.
5. Настройте Apache таким образом, чтобы *DocumentRoot* ссылался на каталог /srv/www/htdocs/squirrelmail.
6. Запустите Apache и с помощью браузера протестируйте результат. Запустите Postfix, Dovecot. Локальные пользователи должны успешно проходить аутентификации, а также иметь возможность отсылать и просматривать электронную почту.

## 21.9 Задание для самопроверки

Настройте свой собственный сервер, использующий данный набор служб:

1. SMTP сервер Postfix (служба *postfix*).
2. Внешний фильтр Amavis (служба *amavisd-new*).
3. Антивирус ClamAV (служба *clamd*)
4. Службу SMTP аутентификации *saslauthd*, входящую в состав Cyrus SASL.
5. POP3/IMAP сервер Dovecot (служба *dovecot*).
6. Почтовый web-интерфейс *SquirrelMail*.
7. Для корректной работы сервера - DNS сервер Bind (служба *named*)
8. Для HTTP доступа к почте - Web сервер Apache (служба *apache2*).



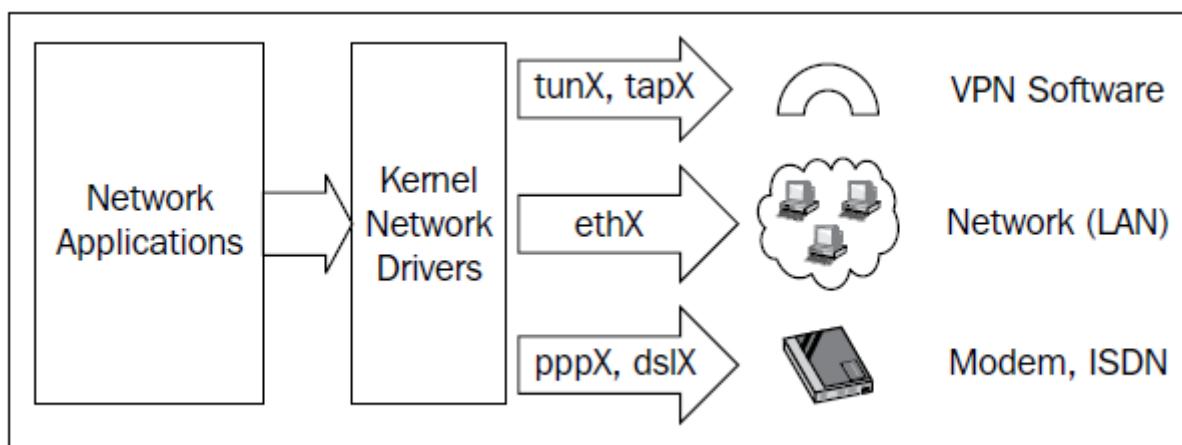
## 22. VPN-сервер на базе OpenVPN

OpenVPN отличается от других VPN решений тем, что использует модульную структуру, которая обеспечивает эффективность OpenVPN технологии при организации безопасных сетевых соединений между удаленным узлами и сетями. В частности, OpenVPN использует универсальный TUN/TAP драйвер, который является open source проектом, используемым во всех современных операционных системах.

Использование TUN/TAP драйвера позволяет упростить структуру OpenVPN. Сложная структура системы – главная беда для ее безопасности, чем проще система, тем меньше уязвимостей в ней может быть найдено.

Универсальный TUN/TAP драйвер разработан для того, чтобы обеспечить ядро Linux поддержкой возможности туннелирования IP трафика. Это виртуальный сетевой интерфейс, который является аутентичным для всех приложений и пользователей. Только имя *tunX* или *tapX* отличает его от других устройств. Каждое приложение, которое имеет возможность взаимодействия с сетевым интерфейсом, может использовать и туннельный интерфейс. Любая сетевая служба, которая используется на сервере, может быть запущена и на TUN или TAP интерфейсах.

Это драйвер является одним из основных факторов того, что делает OpenVPN очень простым для понимания, настройки, и, при этом, очень безопасным. Представим диаграмму, объясняющую использование туннельных сетевых интерфейсов наряду со стандартными сетевыми интерфейсами Linux:



Устройство TUN может рассматриваться как виртуальное соединение точка-точка, как модемное или DSL соединение. Это режим называется *routed mode* ввиду того, что при подключении узлу назначается маршрут к VPN партнерам.

Устройство TAP может рассматриваться как виртуальный Ethernet адаптер, такой режим называется *bridging mode* ввиду того, что сети подключаются друг к другу так, как будто соединены друг с другом аппаратным мостом.

Приложение получают доступ к данным интерфейсам на чтение и запись информации. Драйвер принимает эти данные и шифрует их с использованием криптографических библиотек SSL/TLS. После этого данные отправляются на другую сторону туннеля по UDP (предпочтительнее) или TCP протоколам.

## 22.1 Подготовка к настройке

Перед конфигурированием OpenVPN нам следует проделать несколько шагов:

### Проинсталлируйте OpenVPN

Для инсталляции можно воспользоваться стандартными средствами, например:

```
zypper install openvpn
```

### Выберите *routed* или *bridged* VPN

Для большинства случаев, предпочтительнее использовать *routed* режим, с использованием *tun* интерфейса. Он эффективнее, проще в настройки и предоставляет больше возможностей для селективного контроля доступа клиентов. Режим *bridged* рекомендуется использовать лишь в том случае, если необходимы такие специфичные функции, как: поддержка альтернативных IP протоколов (например, IPX), необходимость использования приложений, использующих широковещательную рассылку (например, какие-либо игры).

### Выберите адрес, который будет использоваться для VPN сети

Настройка VPN сети часто означает соединение друг с другом отдельных сетей, которые могут использовать один из блоков IP адресов, выделенных для частных сетей:

■ Киев	■ Днепропетровск	■ Львов	■ Ровно	■ Мариуполь	■ Полтава
■ Одесса	■ Донецк	■ Николаев	■ Запорожье	■ Луганск	■ Харьков



10.0.0.0	10.255.255.255	(10/8 prefix)
172.16.0.0	172.31.255.255	(172.16/12 prefix)
192.168.0.0	192.168.255.255	(192.168/16 prefix)

Так как адреса из этих сетевых диапазонов должны быть использованы и для настройки VPN сети, очень важно при выборе минимизировать вероятность каких-либо конфликтов. Например, для нашей VPN сети была выбрана сеть 192.168.0.0/24. Когда мы подключаемся к сети из интернет-кафе, которое использует для своей Wi-Fi сети тот же адрес, образуется конфликт в таблице маршрутизации. Наш клиентский компьютер не сможет определить, адрес 192.168.0.1, указанный как шлюз по умолчанию, ссылается на Wi-Fi шлюз по умолчанию или на тот же адрес сети VPN.

Страйтесь избегать адресов типа 10.0.0.0/8 или 192.168.0.0/24. Используйте адреса, которые с наибольшей вероятностью не используются где-либо. Например, 10.40.140.0/24.

## 22.2 Организации PKI инфраструктуры

Прежде чем переходить к настройке OpenVPN, необходимо организовать инфраструктуру PKI (Public Key Infrastructure), которая подразумевает различные сертификаты (публичные ключи) и приватные ключи для сервера и каждого клиента. Кроме того, следует создать сертификат CA (Certificate Authority), которым будут подписаны сертификаты всех узлов VPN сети.

Клиент и сервер для взаимной аутентификации, прежде всего, проверяют, подписан ли представленный для аутентификации сертификат соответствующим СА. Кроме того, используется и дополнительная информация, например, Common Name (CN) сертификата и тип сертификата (клиент или сервер).

Такая модель безопасности имеет ряд положительных моментов. Во-первых, серверу необходима только собственная пара сертификат/частный ключ. Во-вторых, сервер успешно аутентифицирует только тех клиентов, которые представляют





сертификат, подписанный соответствующим СА, а для проверки этого сервер не должен обладать частным ключом СА. Это позволяет хранить данный ключ на другом сервере, что обеспечит его большую защищенность. Это, в свою очередь, приведет к более безопасной модели инфраструктуры PKI, так как в ней самым уязвимым звеном является именно частный ключ СА. В-третьих, в случае потери сертификата, его можно поместить в список CRL (Certificate Revocation List), что позволяет отклонять аутентификацию данного скомпрометированного клиента, без перестройки инфраструктуры PKI. И, наконец, в-четвертых, сервер может проводить контроль доступа на основе клиентских данных, например CN, указанного в сертификате клиента.

Для организации инфраструктуры PKI следует проделать следующие действия:

### **ШАГ 1: Сгенерируйте сертификат и частный ключ Certificate Authority (СА)**

Для управления инфраструктурой PKI удобно использовать набор скриптов, которые поставляются вместе с OpenVPN. Для этого необходимо перейти в каталог, под именем *easy-rsa*, определить местоположение которого на вашей файловой системе можно следующей командой:

```
rpm -qL openvpn | grep easy-rsa
```

```
mail:/usr/share/openvpn/easy-rsa # rpm -qL openvpn | grep easy-rsa
/usr/share/openvpn/easy-rsa
```

Прежде чем выполнять какие-либо скрипты, рекомендуется скопировать содержимое каталога *easy-rsa* (или сам каталог) в какой-либо другой каталог, например */etc/openvpn*. В таком случае, обновление OpenVPN не приведет к изменению настроек, внесенных вами.

После того, как вы скопировали каталог *easy-rsa* и перешли в него, следует отредактировать файл *vars*, в котором обязательно следует задать правильные значения переменным KEY\_COUNTRY, KEY\_PROVINCE, KEY\_CITY, KEY\_ORG и KEY\_EMAIL. Затем необходимо инициализировать инфраструктуру PKI следующим набором команд:

```
./vars
./clean-all
./build-ca
```





Команда `./build-ca` создаст CA сертификат и частный ключ. Процесс генерации выполняется в интерактивном режиме, в ходе которого следует ответить на ряд вопросов:

```
mail:/etc/openvpn/easy-rsa # ./vars
NOTE: when you run ./clean-all, I will be doing a rm -rf on /etc/openvpn/easy-rsa/keys
mail:/etc/openvpn/easy-rsa # ./clean-all
mail:/etc/openvpn/easy-rsa # ./build-ca
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [UA]:
State or Province Name (full name) [OD]:
Locality Name (eg, city) [ODESSA]:
Organization Name (eg, company) [OpenVPN-STEP]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:OpenVPN-STEP-CA
Email Address [root@itstep.org]:
mail:/etc/openvpn/easy-rsa #
```

Обратите внимание, что для большинства запрашиваемых параметров нами уже заданы значения по умолчанию. Единственный параметр, который мы **должны** указать дополнительно – Common Name (CN).

## ШАГ 2: Сгенерируйте сертификат и частный ключ для сервера

Эта задача выполняется следующей командой:

```
./build-key-server server
```

Также как и на предыдущем шаге, большинству параметров нами уже были заданы значения. Исключение опять составляет Common Name – это поле для каждого хоста сети должно быть уникальным. Для сервера необходимо использовать в качестве CN значение *server*. Кроме того, вам будут заданы два вопроса, каждый из которых **требует** положительного ответа ("y"). Первый вопрос: «Подписать ли сертификат?». Второй



вопрос: «Подтвердить ли выпуск нового сертификата?». На оба этих вопроса следует дать утвердительный ответ.

### **ШАГ 3: Сгенерируйте сертификаты клиентов**

Данный шаг почти полностью идентичен предыдущему:

```
./build-key client1  
./build-key client2  
./build-key client3
```

Если необходимы частные ключи, защищенные паролем, то следует использовать сценарий под названием *build-key-pass*.

Не забудьте указывать для каждого клиента уникальное значение параметра *Common Name*. Это является обязательным условием.

### **ШАГ 4: Сгенерируйте параметры алгоритма Диффи-Хеллмана**

Данный алгоритм используется для начального соединения клиентов и сервера. Его параметры генерируются следующим образом:

```
./build-dh
```

### **ШАГ 5: Проверьте наличие сгенерированных ключей и распространите ключи клиентам**

Теперь в подкаталоге *keys* хранятся все сгенерированные нами сертификаты. Для распространения сертификатов и ключей лучше всего воспользоваться SSH протоколом: утилитой *scp*. Если безопасного канала для передачи ключей не существует, следует воспользоваться следующей схемой, в которой частный ключ клиентского узла находится только на жестком диске этого же узла:

- генерируем частный ключ на клиентском узле
- генерируем CSR (Certificate Signing Request)
- отправляем CSR на узел, выполняющий роль СА
- обрабатываем CSR файл, в результате формируется подписанный сертификат узла
- передаем подписанный сертификат на клиентский узел





Приведем список всех ключей и сертификатов, сформированных нами в каталоге *keys*, и дадим им характеристику:

Имя файла	Требуется для	Назначение	Держать в секрете?
<i>ca.crt</i>	Сервер и все клиенты	Корневой CA сертификат	НЕТ
<i>ca.key</i>	Только CA узел	Корневой CA частный ключ	ДА
<i>dh(n).pem</i>	Только сервер	Параметры алгоритма Диффи-Хеллмана	НЕТ
<i>server.crt</i>	Только сервер	Сертификат сервера	НЕТ
<i>server.key</i>	Только сервер	Частный ключ сервера	ДА
<i>client1.crt</i>	Только client1	Сертификат узла client1	НЕТ
<i>client1.key</i>	Только client1	Частный ключ узла client1	ДА
<i>client2.crt</i>	Только client2	Сертификат узла client2	НЕТ
<i>client2.key</i>	Только client2	Частный ключ узла client2	ДА
<i>client3.crt</i>	Только client3	Сертификат узла client3	НЕТ
<i>client3.key</i>	Только client3	Частный ключ узла client4	ДА

## 22.3 Создание конфигурационных файлов сервера и клиентов

Образцы конфигурационных файлов, поставляющиеся в пакете *openvprp*, являются хорошей стартовой точкой для создания собственных настроек. Эти образцы находятся в каталоге *sample-config-files*, который может находиться по адресу */usr/share/doc/packages/openvprp* или */usr/share/doc/openvprp-2.0*.

Образец конфигурационного файла сервера называется *server.conf*, а клиента – *client.conf*. Рассмотрим подробнее каждый из данных файлов.

### 22.3.1 Конфигурация сервера

Для настройки сервера скопируйте файл *server.conf* в каталог, в котором будут храниться все необходимые данные для работы сервера. Это, например, может быть каталог */etc/openvprp*, в котором мы на предыдущем этапе создали каталог *keys*, содержащий ключи сервера.





После копирования файла *server.conf* следует отредактировать его параметры. Набор конфигураций заданный по умолчанию уже обладает достаточной функциональностью для того, чтобы запустить VPN сервер с помощью TUN интерфейса. Сам сервер будет использовать UDP порт 1194, который является стандартным для OpenVPN. Адреса, которые будут распределяться клиентам, принадлежат сети 10.8.0.0/24. Таким образом, для запуска сервера достаточно исправить лишь несколько параметров.

Первым делом необходимо указать пути к файлам сертификатов и ключей, в частности, задать правильные значения таким параметрам, как: *ca*, *cert*, *key* и *dh*. Например:

```
# Any X509 key management system can be used.  
# OpenVPN can also use a PKCS #12 formatted key file  
# (see "pkcs12" directive in man page).  
ca keys/ca.crt  
cert keys/server.crt  
key keys/server.key # This file should be kept secret  
  
# Diffie hellman parameters.  
# Generate your own with:  
# openssl dhparam -out dh1024.pem 1024  
# Substitute 2048 for 1024 if you are using  
# 2048 bit keys.  
dh keys/dh1024.pem
```

В нашем примере пути к сертификатам и ключам указаны относительно каталога, в котором находится файл *server.conf*.

Если мы хотим вместо TUN интерфейса использовать ТАР интерфейс, необходимо внести следующие корректизы в конфигурационном файле: вместо *server* использовать опцию *server-bridge* и вместо *dev tun* использовать опцию *dev tap*.

Если есть необходимость в том, чтобы использовать TCP вместо UDP – следует исправить опцию *proto udp* на значение *proto tcp*.

Следующий этап – адрес сети, который будет использоваться нашей сетью VPN. Неиспользование «стандартных» адресов сети является одним из условий корректной настройки сети VPN, данное правило мы сформулировали в одном из предыдущих разделов. Для того чтобы задать необходимый адрес, следует редактировать аргументы параметра *server*:



```
# Configure server mode and supply a VPN subnet
# for OpenVPN to draw client addresses from.
# The server will take 10.8.0.1 for itself,
# the rest will be made available to clients.
# Each client will be able to reach the server
# on 10.8.0.1. Comment this line out if you are
# ethernet bridging. See the man page for more info.
server 10.40.0.0 255.255.255.0
```

Если наша сеть VPN нуждается в том, чтобы клиенты могли взаимодействовать друг с другом, а не только с сервером, следует раскомментировать директиву *client-to-client*.

После инициализации службы OpenVPN можно понизить ее полномочия таким образом, чтобы в дальнейшем все операции проводились от имени пользователя *nobody* и одноименной группы:

```
# It's a good idea to reduce the OpenVPN
# daemon's privileges after initialization.
#
# You can uncomment this out on
# non-Windows systems.
user nobody
group nobody
```

### 22.3.2 Конфигурация клиента

Образцовый файл конфигурации *client.conf* также практически готов для использования с теми параметрами, которые заданы в нем по умолчанию. Аналогично файлу *server.conf* следует внести туда несколько обязательных и необязательных изменений.

К обязательным параметрам относятся *ca*, *cert* и *key*. Каждый настраиваемый клиент должен обязательно иметь свою собственную, уникальную пару *cert/key*. Только *ca* файл одинаковый для сервера и клиентов настраиваемой нами инфраструктуры.

Также следует отредактировать директиву *remote*, в которой необходимо указать IP адрес или DNS имя VPN сервера, к которому следует подключаться клиенту. Кроме того, необходимо следить за тем, чтобы значения некоторых параметров в конфигурационных файлах клиентов и сервера совпадали. В частности, это относится к



таким параметрам: *dev* (*tun* или *tap*), *proto* (*udp* или *tcp*), *comp-lzo* (задан или не задан) и *fragment* (задан или не задан).

## 22.4 Запуск и тестирование VPN соединения

Перед запуском сервера следует убедить в том, что VPN сервер доступен для клиентов. Если речь идет о взаимодействии клиентов и сервера в рамках сети Интернет, то возможны два сценария:

- Сервер имеет публичный адрес, и правила брандмауэра не запрещают доступ к порту, на котором запущен OpenVPN сервер.
- Сервер имеет адрес в частной сети. В таком случае необходимо настроить переадресацию портов со шлюза, через который OpenVPN сервер получает доступ в сеть Интернет.

Затем, следует запустить сервер:

```
mail:~/Desktop # ls /etc/openvpn/keys/
01.pem  ca.crt      client1.csr  client2.csr  index.txt          index.txt.old  server.crt
02.pem  ca.key      client1.key  client2.key  index.txt.attr    serial        server.csr
03.pem  client1.crt client2.crt  dh1024.pem   index.txt.attr.old serial.old    server.key
mail:~/Desktop # cd /etc/openvpn/
mail:/etc/openvpn # openvpn server.conf
Mon Jan 24 10:32:14 2011 OpenVPN 2.0.9 i586-suse-linux [SSL] [LZO] built on Feb 25 2009
Mon Jan 24 10:32:14 2011 MANAGEMENT: TCP Socket listening on 127.0.0.1:7505
Mon Jan 24 10:32:14 2011 Diffie-Hellman initialized with 1024 bit key
Mon Jan 24 10:32:14 2011 TLS-Auth MTU parms [ L:1542 D:138 EF:38 EB:0 ET:0 EL:0 ]
Mon Jan 24 10:32:14 2011 TUN/TAP device tun0 opened
Mon Jan 24 10:32:14 2011 /bin/ip link set dev tun0 up mtu 1500
Mon Jan 24 10:32:14 2011 /bin/ip addr add dev tun0 local 10.40.0.1 peer 10.40.0.2
Mon Jan 24 10:32:14 2011 /bin/ip route add 10.40.0.0/24 via 10.40.0.2
Mon Jan 24 10:32:14 2011 Data Channel MTU parms [ L:1542 D:1450 EF:42 EB:135 ET:0 EL:0 AF:3/1 ]
Mon Jan 24 10:32:14 2011 GID set to nobody
Mon Jan 24 10:32:14 2011 UID set to nobody
Mon Jan 24 10:32:14 2011 UDPv4 link local (bound): [undef]:1194
Mon Jan 24 10:32:14 2011 UDPv4 link remote: [undef]
Mon Jan 24 10:32:14 2011 MULTI: multi_init called, r=256 v=256
Mon Jan 24 10:32:14 2011 IFCONFIG POOL: base=10.40.0.4 size=62
Mon Jan 24 10:32:14 2011 IFCONFIG POOL LIST
Mon Jan 24 10:32:14 2011 client,10.40.0.4
Mon Jan 24 10:32:14 2011 Initialization Sequence Completed
```

Сервер успешно запущен и можно приступать к запуску клиента:

*openvpn файл\_конфигурации\_клиента*

■ Киев	■ Днепропетровск	■ Львов	■ Ровно	■ Мариуполь	■ Полтава
■ Одесса	■ Донецк	■ Николаев	■ Запорожье	■ Луганск	■ Харьков





После этого следует проверить связь с сервером командой *ping*. В качестве аргумента команды используйте IP адрес сервера в сети VPN. Это всегда первый адрес в той сети, которая была выделена нами для VPN. В нашем примере, адрес сервера 10.40.0.1:

```
ping 10.40.0.1
```

Если *ping* прошел успешно – сеть VPN работает!

## 22.5 Наблюдение за сервером

По умолчанию файл конфигурации сервера содержит следующую строку:

```
status openvpn-status.log
```

Каждую минуту данный файл обновляется информацией о клиентах, подключенных к серверу.

Еще один способ наблюдения (и даже управления) – использование интерфейса управления OpenVPN, который запускается при наличии следующей директивы в конфигурационном файле сервера:

```
management localhost 7505
```

Данная директива указывает, что управляющий интерфейс будет запущен на TCP порте 7505 текущего узла.

Для подключения к серверу следует использовать утилиту *telnet*:

```
mail:/etc/openvpn # telnet 127.0.0.1 7505
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^].
>INFO:OpenVPN Management Interface Version 1 -- type 'help' for more info
```

Для получения справки про команды, которые можно использовать в данном интерфейсе, используйте команду *help*. В представленном листинге команд вы найдете способ наблюдения за клиентами, получите возможность отключения клиентов и получите информацию о других полезных действиях:



```

mail:/etc/openvpn # telnet 127.0.0.1 7505
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^].
>INFO:OpenVPN Management Interface Version 1 -- type 'help' for more info
help
Management Interface for OpenVPN 2.0.9 i586-suse-linux [SSL] [LZO] [EPOLL] built on Feb 25 2009
Commands:
auth-retry t          : Auth failure retry mode (none,interact,no interact).
echo [on|off] [N|all] : Like log, but only show messages in echo buffer.
exit|quit             : Close management session.
help                 : Print this message.
hold [on|off|release]: Set/show hold flag to on/off state, or
                      release current hold and start tunnel.
kill cn               : Kill the client instance(s) having common name cn.
kill IP:port          : Kill the client instance connecting from IP:port.
log [on|off] [N|all]  : Turn on/off realtime log display
                      + show last N lines or 'all' for entire history.
mute [n]              : Set log mute level to n, or show level if n is absent.
net                  : (Windows only) Show network info and routing table.
password type p       : Enter password p for a queried OpenVPN password.
signal s              : Send signal s to daemon,
                      s = SIGHUP|SIGTERM|SIGUSR1|SIGUSR2.
state [on|off] [N|all] : Like log, but show state history.
status [n]             : Show current daemon status info using format #n.
test n                : Produce n lines of output for testing/debugging.
username type u       : Enter username u for a queried OpenVPN username.
verb [n]               : Set log verbosity level to n, or show if n is absent.
version              : Show current version number.
END

```

При тестировании соединений важно знать, что при использовании TUN интерфейса OpenVPN выдает адреса для сервера и клиентов следующим образом (применительно к нашему примеру):

Адрес сервера: 10.40.0.1 point-to-point соединение с адресом 10.40.0.2  
 Адрес 1-го клиента: 10.40.0.6 point-to-point соединение с адресом 10.40.0.5  
 Адрес 2-го клиента: 10.40.0.10 point-to-point соединение с адресом 10.40.0.9  
 Адрес 3-го клиента: 10.40.0.14 point-to-point соединение с адресом 10.40.0.13  
 И так далее.

## 22.6 Задание для самопроверки

1. Настройте OpenVPN сервер и подключите к нему трех клиентов таким образом, чтобы они могли взаимодействовать друг с другом.



## 23. Маршрутизация

IP-маршрутизацией называется процесс, при помощи которого машина с несколькими сетевыми интерфейсами выбирает, через какой из интерфейсов послать полученный IP-пакет.

Проиллюстрируем это определение на примере. Представьте себе обычный офисный маршрутизатор, который имеет PPP-соединение с интернетом, несколько подключенных сегментов локальной ethernet-сети и PPP-соединение с другим офисом.

Когда через один из интерфейсов на маршрутизатор приходит IP-пакет, маршрутизатор должен решить, через какой из интерфейсов отправлять этот пакет далее. Даже простым машинам может потребоваться маршрутизировать пакеты, так как они имеют, по меньшей мере, два интерфейса -- один кольцевой (loopback), и второй, через который они подключены к реальной сети, например ethernet-интерфейс, или интерфейс подключения по последовательным линиям PPP.

Как же происходит маршрутизация? На каждой машине хранится специальный список правил маршрутизации, называемый *таблицей маршрутизации*. Любая строка этой таблицы, как правило, содержит, по меньшей мере, три поля. Первое поле - адрес назначения, второе - имя интерфейса, через который следует отправлять пакеты, и третье - IP-адрес машины, через которую будут передаваться пакеты. В Linux, вы можете просмотреть таблицу маршрутизации с помощью следующей команды:

```
cat /proc/net/route
```

или одной из команд:

```
route -n  
netstat -r
```

Процесс маршрутизации достаточно прост: когда машина получает IP-пакет, его адрес назначения (адрес машины, на которую отправлен этот пакет) сверяется с каждой строкой таблицы маршрутизации. Выбирается подходящая строка и пакет передается через указанный в этой строке интерфейс. Если поле адреса `машины-передатчика` заполнено, то пакет передается на эту машину, иначе считается, что адресат пакета находится в сети, к которой подключен данный интерфейс.





Для управления таблицей маршрутизации используется программа `route`. Эта программа преобразует свои аргументы в параметры вызова ядра, и ядро добавляет, удаляет или изменяет строки в таблице маршрутизации.

Простой пример. Представьте себе, что у вас есть Ethernet сеть. Она организована как сеть класса С с адресом '192.168.1.0'. Вашей машине был выделен адрес '192.168.1.10' и было сказано, что маршрутизатор, через который Ваша сеть подключена к интернету, находится по адресу '192.168.1.1'.

Первым делом Вы должны настроить сетевой интерфейс:

```
ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
```

После этого Вы должны добавить в таблицу маршрутизации на Вашей машине строку, согласно которой пакеты на машины с адресами '192.168.1.\*' ядро должно отправлять через интерфейс eth0. Это делается с помощью следующей команды:

```
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```

Обратите внимание на использование опции `-net`. Эта опция указывает, что адрес назначения в таблице маршрутизации будет адресом сети. С помощью опции `-host` вы можете задать маршрут на конкретный IP адрес.

Этот маршрут позволит вам устанавливать IP соединения со всеми машинами в вашем локальном Ethernet сегменте. Но как быть со всеми остальными машинами?

Было бы очень сложно задавать маршруты для всех возможных IP-сетей явно, поэтому используют следующий трюк - маршрут по умолчанию. Маршрут по умолчанию подходит для всех адресов назначения, не указанных в таблице маршрутизации. С помощью маршрута по умолчанию Вы говорите ядру - "а все остальное отправляй туда". В нашем примере маршрут по умолчанию настраивается командой:

```
route add default gw 192.168.1.1 eth0
```

Опция `gw` указывает программе *route* что следующий аргумент – IP адрес или имя маршрутизатора, на который надо отправлять все пакеты, соответствующие этой строке таблицы маршрутизации.





Итак, полностью настройка будет выглядеть так:

```
ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
route add default gw 192.168.1.1 eth0
```

Рассмотрим теперь несколько более сложную конфигурацию маршрутизации в сети. Представьте себе, что вы должны настроить маршрутизатор из предыдущего примера. У этого маршрутизатора есть одно PPP-соединение и три подключенных Ethernet сегмента. Настройка маршрутизации будет выглядеть так:

```
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
route add -net 192.168.2.0 netmask 255.255.255.0 eth1
route add -net 192.168.3.0 netmask 255.255.255.0 eth2
route add default ppp0
```

Во всей сети только маршрутизатор должен иметь в своей таблице маршрутизации отдельные строки для каждой из подсетей. На остальных машинах связь с другими сегментами локальной сети будет осуществляться с помощью маршрута по умолчанию. Они будут отправлять пакеты на маршрутизатор, о тот будет передавать их в нужный сегмент сети. Вас может удивить, что маршрут по умолчанию на маршрутизаторе не использует опции '*gw*'. Причина проста. Протоколы соединения по последовательным линиям, такие как PPP и SLIP всегда имеют только две машины в сети - (соединение точка-точка) поэтому указание адреса бессмысленно - на том конце соединения только одна машина. Таким образом, для подобных соединений нет нужды указывать адрес маршрутизатора, на который надо передавать пакеты. Для других типов сетей, таких как *Ethernet* или *token ring* требуется указывать адрес маршрутизатора, так как эти сети поддерживают подключение сразу многих машин к одному сегменту сети.

Настройка маршрутизации в приведенных примерах лучше всего подходит для простых сетей, в которых между точками отправки и назначения существует единственный маршрут. В более сложных сетях это не так, и маршрутизация становится сложнее.

В данной главе нами будут рассмотрены расширенные возможности по настройке статической маршрутизации, а также основы настройки динамической маршрутизации, использование которой позволяет облегчить труд системного администратора.

## 23.1 Iproute2

- █ Киев
- █ Днепропетровск
- █ Львов
- █ Ровно
- █ Мариуполь
- █ Полтава
- █ Одесса
- █ Донецк
- █ Николаев
- █ Запорожье
- █ Луганск
- █ Харьков





Набор инструментов, которые входят в состав пакета *iproute2*, позволяют повысить эффективность управления трафиком в системах, построенных на базе Linux. Широко известные команды, такие как *route* и *ifconfig* - фактически являются довольно тонкой оберткой вокруг очень мощной инфраструктуры *iproute2*. Вот далеко неполный список из того, что может предложить вам операционная система Linux:

- Управлять пропускной способностью на отдельных компьютерах.
- Управлять пропускной способностью к отдельным компьютерам.
- Поможет "раздать" пропускную способность "по справедливости".
- Защитить вашу сеть от DoS-атак.
- Предотвратить нападения из вашей сети на серверы в Интернет.
- Распараллелить несколько серверов, с целью равномерного распределения нагрузки.
- Ограничить доступ к вашим компьютерам.
- Ограничить доступ ваших пользователей к другим узлам сети.
- Выполнять маршрутизацию на основе UID, MAC-адресов, исходящих IP адресов, номеров портов, типа обслуживания, времени суток и содержимого.

В данном разделе мы рассмотрим некоторые из описанных выше возможностей. Для использования всех этих возможностей следует установить пакет *iproute2*:

```
zypper install iproute2
```

### 23.1.1 Текущая конфигурация

Сразу же после инсталляции *iproute2* уже сконфигурирована. Утилита *ip* является основной в пакете. Попробуем с ее помощью исследовать имеющиеся в системе сетевые интерфейсы.

#### 23.1.1.1 Просмотр списка сетевых интерфейсов

Рассмотрим результат работы команды *ip link list* (у вас он может несколько отличаться):





```
mail:/etc/iproute2 # ip link list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:7c:eb:56 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:d5:37:f7 brd ff:ff:ff:ff:ff:ff
```

Первым в списке находится локальный (loopback) интерфейс. Размер максимального блока передачи данных (MTU - Maximum Transfer Unit) для этого интерфейса составляет 16436 байт, и для него отсутствует очередь, поскольку он не соответствует никакому физическому устройству и существует только в "воображении" ядра.

Дальше следуют два физических сетевых интерфейса. Обратите внимание на отсутствие IP-адресов в листинге. Понятие "интерфейса" отделяет в *iproute2* от понятия "IP-адреса". При назначении нескольких IP адресов одному интерфейсу (IP алиасинг), понятие IP-адреса становится достаточно расплывчатым. Зато показываются MAC-адреса - аппаратные идентификаторы сетевых интерфейсов.

### 23.1.1.2 Просмотр списка IP адресов

Для просмотра списков IP адресов используется команда *ip address show*:

```
mail:/etc/iproute2 # ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
            netmask v4掩码
            broadcast v4广播地址
        inet 127.0.0.2/8 brd 127.255.255.255 scope host secondary lo
            netmask v4掩码
            broadcast v4广播地址
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:7c:eb:56 brd ff:ff:ff:ff:ff:ff
        inet 10.0.6.220/27 brd 10.0.6.223 scope global eth0
            netmask v4掩码
            broadcast v4广播地址
        inet 192.168.40.1/24 brd 192.168.40.255 scope global eth1
            netmask v4掩码
            broadcast v4广播地址
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:d5:37:f7 brd ff:ff:ff:ff:ff:ff
        inet 192.168.40.1/24 brd 192.168.40.255 scope global eth1
            netmask v4掩码
            broadcast v4广播地址
```

Этот листинг содержит более подробную информацию. Здесь показаны все IP-адреса, и каким интерфейсам они принадлежат. Здесь "inet" соответствует термину "Internet (IPv4)". Существует целый ряд типов сетевых адресов, но нас они пока не интересуют.





Взглянем поближе на интерфейс eth1. Из листинга видно, что ему назначен адрес "inet" – 192.168.40.1/24, где "/24" определяет число бит, соответствующих адресу сети. Таким образом, для адресации хостов в сети у нас остается  $32 - 24 = 8$  бит, что соответствует адресу сети – 192.168.40.0 и маске сети - 255.255.255.0.

Это говорит о том, что любой хост в этой сети, например 192.168.40.100, будет непосредственно доступен через наш интерфейс с IP-адресом 192.168.40.1.

Для eth0 применима та же концепция, хотя числа в IP-адресе отличаются. Попробуйте самостоятельно вычислить адрес сети данного узла.

### 23.1.1.3      Просмотр списка маршрутов

Для просмотра списков маршрутов используется команда *ip route show*:

```
mail:~/Desktop # ip route show
10.0.6.192/27 dev eth0 proto kernel scope link src 10.0.6.220
192.168.40.0/24 dev eth1 proto kernel scope link src 192.168.40.1
169.254.0.0/16 dev eth0 scope link
127.0.0.0/8 dev lo scope link
default via 10.0.6.193 dev eth0
```

Этот листинг достаточно "прозрачен". Первые две строки сообщают сведения, которые нами уже обсуждались выше. Третья строка задает маршрут в сеть APIPA, четвертая описывает loopback, а последняя строка говорит о том, что внешний мир доступен через 10.0.6.193 - шлюз, заданный по умолчанию. То, что это шлюз, видно благодаря наличию слова "via" (в переводе с англ. - "через"). Этот шлюз (с адресом 10.0.6.193) готов перенаправлять наши пакеты в сети, чьи адреса явно не указаны в предыдущих строках таблицы маршрутизации.

Для примера, более "старая" утилита *route*, дает такой результат:

```
mail:~/Desktop # route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
10.0.6.192      0.0.0.0        255.255.255.224 U        0      0        0 eth0
192.168.40.0    0.0.0.0        255.255.255.0   U        0      0        0 eth1
169.254.0.0     0.0.0.0        255.255.0.0    U        0      0        0 eth0
127.0.0.0       0.0.0.0        255.0.0.0     U        0      0        0 lo
0.0.0.0          10.0.6.193    0.0.0.0      UG       0      0        0 eth0
```



## 23.1.2 ARP

ARP - Address Resolution Protocol (Протокол Определения Адреса) описан в RFC 826. Он используется для определения Ethernet адреса (MAC адрес) по IP адресу. Машины в Интернет более известны под именами, которые преобразуются в IP-адреса, благодаря чему узел сети, скажем с именем foo.com, имеет возможность связаться с другой машиной, например с именем bar.net. Но в Ethernet сетях для адресации используется не IP адрес, а Ethernet адрес и здесь на сцену выходит протокол ARP.

Рассмотрим простой пример. Предположим, что имеется сеть из нескольких компьютеров. В ней находятся компьютеры foo, с адресом 10.0.0.1 и foo, с адресом 10.0.0.2. Пусть foo хочет послать пакет ICMP Echo Request (*ping*) компьютеру bar, чтобы проверить - работает ли он, но увы, foo не знает ethernet-адреса компьютера bar. Таким образом, прежде чем "ping-ануть" bar, foo должен отослать ARP-запрос. Этот запрос очень похож на то, что обычно кричит человек, пытаясь отыскать в толпе своего товарища: "Bar (10.0.0.2)! Ты где?". В результате все машины в сети услышат "крик" foo, но только bar (10.0.0.2) откликнется на него, послав обратно ARP-ответ, который можно трактовать как: "Foo (10.0.0.1)! Я - здесь! Мой адрес 00:60:94:E9:08:12.". После этой "переклички" foo будет знать Ethernet адрес компьютера bar и сможет связаться с ним, пока опять не "забудет" (в кэше ARP) адрес компьютера bar (обычно записи в ARP-кэше удаляются через 15 минут).

Содержимое ARP-кэша можно просмотреть так:

```
mail:/etc/sysconfig/network/scripts # ip neighbour show
10.0.6.193 dev eth0 lladdr 00:15:e9:2f:33:e4 REACHABLE
10.0.6.196 dev eth0 lladdr 08:00:27:1c:e4:8a STALE
```

## 23.1.3 Правила – база политик маршрутизации

Если маршрутизатор обслуживает сложную сеть, то нужно удовлетворять нужды разных людей, обслуживание которых, вероятно, должно отличаться. База политик маршрутизации позволяет реализовать это с помощью набора таблиц маршрутизации. Если вы хотите использовать эту возможность, убедитесь что ядро собрано с поддержкой "IP: advanced router" и "IP: policy routing".

Когда ядру необходимо выбрать маршрут, оно определяет в соответствии с какой таблицей это нужно делать. По умолчанию, определены три таблицы. Старая утилита





*route* изменяет таблицы *main* и *local*, как и утилита *ip* (по-умолчанию). Правила по-умолчанию:

```
[root@mail ~]# ip rule list
0: from all lookup Local
32766: from all lookup main
32767: from all lookup default
```

В этом листинге приведены приоритеты всех правил. Мы видим, что правила применяются ко всем пакетам (*from all*). Мы уже видели таблицу '*main*', она выводится командой *ip route ls*, но таблицы '*local*' и '*default*' для нас новые.

Если мы хотим сделать что-то интересное, то нужно задать правила, использующие разные таблицы маршрутизации. Это позволит нам переопределить общесистемную таблицу маршрутизации.

### 23.1.3.1 Простая маршрутизация – по источнику

Давайте опять рассмотрим реальный пример. Например, у нас есть 2 кабельных модема, подключенных к маршрутизатору Linux с NAT ('masquerading'). Люди с которыми мы живем в одном доме, платят нам за использование Internet. Допустим, одни из соседей ходят только на hotmail и хочет платить меньше. Мне это подходит, но при этом будет использоваться медленный канал.

Быстрое соединение имеет с нашей стороны адрес 212.64.94.251, а с другой - 212.64.94.1. Медленное соединение получает динамический адрес, в данном примере это 212.64.78.148, адрес провайдера - 195.96.98.253. Посмотрим на таблицу *local*:

```
[root@mail ~]# ip route list table Local
broadcast 127.255.255.255 dev lo proto kernel scope link src 127.0.0.1
Local 10.0.0.1 dev eth0 proto kernel scope host src 10.0.0.1
broadcast 10.0.0.0 dev eth0 proto kernel scope link src 10.0.0.1
Local 212.64.94.251 dev ppp0 proto kernel scope host src 212.64.94.251
broadcast 10.255.255.255 dev eth0 proto kernel scope link src 10.0.0.1
broadcast 127.0.0.0 dev lo proto kernel scope link src 127.0.0.1
Local 212.64.78.148 dev ppp2 proto kernel scope host src 212.64.78.148
Local 127.0.0.1 dev lo proto kernel scope host src 127.0.0.1
Local 127.0.0.0/8 dev lo proto kernel scope host src 127.0.0.1
```

Посмотрим теперь на таблицу *main* :





```
[root@mail ~]# ip route list table main
195.96.98.253 dev ppp2 proto kernel scope link src 212.64.78.148
212.64.94.1 dev ppp0 proto kernel scope link src 212.64.94.251
10.0.0.0/8 dev eth0 proto kernel scope link src 10.0.0.1
127.0.0.0/8 dev lo scope link
default via 212.64.94.1 dev ppp0
```

Создадим новое правило для нашего гипотетического соседа, которое будет называться 'John'. Хотя мы можем работать просто с числами, намного проще и понятней, если мы определим названия наших таблиц в файле `/etc/iproute2/rt_tables`.

```
[root@mail ~]# echo 200 John >> /etc/iproute2/rt_tables
[root@mail ~]# ip rule add from 10.0.0.10 table John
[root@mail ~]# ip rule ls
0:from all Lookup Local
32765: from 10.0.0.10 Lookup John
32766: from all Lookup main
32767: from all Lookup default
```

Теперь нам нужно лишь сгенерировать таблицу John и очистить кэш маршрутов:

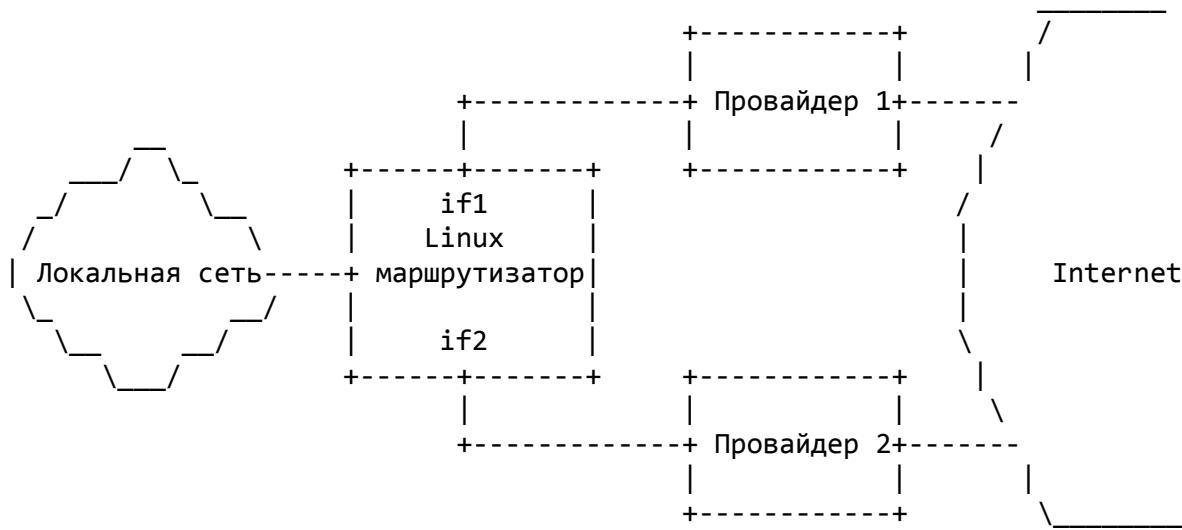
```
[root@mail ~]# ip route add default via 195.96.98.253 dev ppp2 table John
[root@mail ~]# ip route flush cache
```

Вот и все. Вы можете добавить это в скрипт инициализации сетевых интерфейсов и использовать.



## 23.1.4 Маршрутизация через несколько каналов

Ниже представлена обычная конфигурация, когда локальная сеть (или даже одна машина) подключена к Internet через двух провайдеров.



Рассмотрим, как в такой конфигурации организовать раздельный доступ и распределение нагрузки.

### 23.1.4.1 Раздельный доступ

Первый вопрос заключается в том, как организовать маршрутизацию таким образом, чтобы ответы на запросы, приходящие через определенного провайдера, скажем провайдера 1, уходили через того же провайдера.

Давайте определим некоторые переменные. Пусть \$IF1 будет именем первого интерфейса (if1 на рисунке), а \$IF2 - именем второго. Тогда \$IP1 будет IP адресом \$IF1, а \$IP2 - IP адресом \$IF2. Далее, \$P1 это IP-адрес шлюза провайдера 1, а \$P2 - IP адрес шлюза провайдера 2. Наконец, \$P1\_NET это IP сеть, к которой принадлежит \$P1, а \$P2\_NET - сеть, к которой принадлежит \$P2.

Создадим две дополнительные таблицы маршрутизации, скажем T1 и T2. Добавим их в файл /etc/iproute2/rt\_tables. Теперь можно настроить эти таблицы следующими командами:



```
ip route add $P1_NET dev $IF1 src $IP1 table T1
ip route add default via $P1 table T1
ip route add $P2_NET dev $IF2 src $IP2 table T2
ip route add default via $P2 table T2
```

Ничего особо эффектного, маршрут к шлюзу и маршрут по умолчанию через этот шлюз. Точно так же, как и в случае одного провайдера, но по таблице на каждого провайдера. Заметьте, что маршрута к сети, в которой находится шлюз достаточно, потому что он определяет, как найти все хосты в этой сети, включая сам шлюз.

Теперь нужно настроить главную таблицу маршрутизации. Хорошо бы маршрутизировать пакеты для сетей провайдеров через соответствующие интерфейсы. Обратите внимание на аргумент 'src', который обеспечивает правильный выбор исходного IP-адреса.

```
ip route add $P1_NET dev $IF1 src $IP1
ip route add $P2_NET dev $IF2 src $IP2
```

Теперь задаем маршрут по умолчанию:

```
ip route add default via $P1
```

Зададим правила маршрутизации. Они будут отвечать за то, какая таблица будет использоваться при маршрутизации. Вы хотите, чтобы пакет с определенным адресом источника маршрутизировался через соответствующий интерфейс:

```
ip rule add from $IP1 table T1
ip rule add from $IP2 table T2
```

Этот набор команд обеспечивает маршрутизацию ответов через интерфейс, на котором был получен запрос.

Итак, мы рассмотрели очень простой пример. Он будет работать для всех процессов, выполняющихся на маршрутизаторе и для локальной сети, если настроено преобразование адресов (NAT/masquerading). В противном случае, вам будет необходим диапазон IP адресов обоих провайдеров, или выполнять маскирование для одного из провайдеров. В любом случае, вы можете задать правила выбора провайдера для каждого конкретного адреса вашей локальной сети.



### 23.1.4.2 Распределение нагрузки

Второй вопрос заключается в балансировке нагрузки между двумя провайдерами. Это не сложно, если у вас уже настроен раздельный доступ, описанный в предыдущем разделе.

Вместо выбора одного из провайдеров в качестве маршрута по умолчанию, вы настраиваете т.н. многолучевой (multipath) маршрут. В стандартном ядре это обеспечит балансировку нагрузки между двумя провайдерами. Делается это следующим образом (повторюсь, мы основываемся на примере из раздела Раздельный доступ):

```
ip route add default scope global nexthop via $P1 dev $IF1 weight 1 nexthop \
via $P2 dev $IF2 weight 1
```

Результатом команды будет попеременный выбор маршрута по-умолчанию. Вы можете изменить параметр `weight`, так чтобы один из провайдеров получал большую нагрузку.

## 23.2 Quagga

Установив пакет *quagga*, вы с легкостью превратите Linux систему в мощный маршрутизатор, использующий в своей работе такие протоколы динамической маршрутизации как RIP, OSPF и BGP.

Благодаря Quagga наша система может обмениваться информацией о маршрутизации с другими маршрутизаторами. Quagga позволяет динамически обновлять конфигурацию с помощью специального терминального интерфейса, подключение к которому осуществляется через *telnet*. При настройке Quagga используется синтаксис, присущий Cisco IOS. Если вам приходилось работать с Cisco IOS, вы без труда освоите Quagga.

Использование протоколов динамической маршрутизации с помощью Quagga существенно облегчает работу администратора, так как в сетях, которые подвержены постоянным модификациям, статическая маршрутизация малопригодна.

Quagga работает в двух пользовательских режимах. В нормальном режиме пользователь может просматривать состояние системы, маршруты и настройки



интерфейсов. В привилегированном режиме, который запускается командой *enable*, можно редактировать настройки и обновлять параметры конфигурации. Такая модель, независящая от локальных Linux пользователей, помогает администратору в делегировании полномочий по обслуживанию маршрутизатора.

Можно смело говорить о том, что Quagga – достаточно производительное программное обеспечение, которое позволяет администратору организовать эффективную и качественную маршрутизацию, с использованием лишь open source решений.

Для инсталляции, достаточно воспользоваться традиционной командой:

```
zypper install quagga
```

### 23.2.1 Архитектура Quagga

Quagga – это набор из нескольких служб, каждая из которых решается свою собственную задачу.

Служба *ripd* – обслуживает протокол RIP, служба *ospfd* – протокол OSPF, *bgpd* – BGP. Для изменения таблиц маршрутизации ядра Linux и для распределения маршрутов между различными протоколами маршрутизации, используется служба *zebra*, которая обслуживает таблицы маршрутизации ядра. Такая архитектура Quagga позволяет с легкостью добавлять новые службы, которые будут обслуживать новые, еще не поддерживаемые в Quagga, протоколы маршрутизации.

Каждая служба имеет свой собственный конфигурационный файл. Для настройки статической маршрутизации следует редактировать конфигурационный файл службы *zebra*, а для настройки протокола RIP – конфигурационный файл службы *ripd*. Это может быть неудобным, поэтому в Quagga встроена возможность конфигурирования с помощью специального интерфейса, который называется *vtysh*.

Вы можете заранее готовить конфигурационные файлы служб, или можете подключаться к пользовательскому интерфейсу с помощью *telnet* и настраивать маршрутизацию в интерактивном режиме. Используйте просто наиболее удобный для вас вариант, итоговый результат всегда будет одинаковый.



## 23.2.2 Zebra

Конфигурационные файлы Quagga хранятся в каталоге */etc/quagga*. Для успешной работы Quagga, в данном каталоге необходимо, как минимум, наличие файла *zebra.conf*, в котором задаются настройки соответствующей службы. Рассмотрим простейшую настройку файла */etc/quagga/zebra.conf*:

```
hostname quagga          # hostname маршрутизатора
password quagga          # пароль доступа к пользовательскому интерфейсу
enable password quagga   # пароль доступа к привилегированному режиму
Log file /var/Log/quagga/quagga.Log # адрес журнального файла
```

Запустим службу *zebra* и подключимся с помощью *telnet* к пользовательскому терминалу службы *zebra*. Для этого обратимся на TCP порт под номером 2601:

```
mail:~/Desktop # rczebra start
Starting routing daemon (Zebra)                                         done
mail:~/Desktop # telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^']'.

Hello, this is Quagga (version 0.99.15).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification

Password:
quagga>
  echo      Echo a message back to the vty
  enable    Turn on privileged mode command
  exit      Exit current mode and down to previous mode
  help      Description of the interactive help system
  list      Print command list
  quit      Exit current mode and down to previous mode
  show      Show running system information
  terminal  Set terminal line parameters
  who       Display who is on vty
quagga> ■
```

Для получения справки в пользовательском интерфейсе Quagga, следует использовать символ “?”, на изображении виден результат ввода данного символа – получен список команд, доступных на текущем уровне конфигурирования.



### 23.2.3 Ripd

RIP маршрутизатор обменивается информацией о маршрутах со своими соседними маршрутизаторами, что позволяет хранить актуальные таблицы маршрутизации на всех RIP маршрутизаторах сети. Для обмена информацией RIP использует UDP 520 порт, который должен быть «открыт» соответствующими правилами вашего брандмауэра.

Прежде чем запускать *ripd*, следует запускать *zebra*. Кроме того, необходимо создать конфигурационный файл */etc/quagga/ripd.conf*, в котором, как минимум, следует задать опцию *password*. А для того, чтобы запустить вполне функциональный RIP маршрутизатор, достаточно использовать следующую конфигурацию:

```
hostname rip          # hostname маршрутизатора
password rip          # пароль доступа к пользовательскому интерфейсу
router rip            # включаем RIP
version 2              # используем RIPv2
network eth0          # для работы протокола используется сетевой интерфейс eth0
redistribute connected # анонсируем маршруты к тем сетям, которые подключены к роутеру
```

После настройки *ripd.conf* следует запустить службу *ripd*. Для доступа к интерфейсу конфигурирования, необходимо обратиться с помощью *telnet* на TCP 2602 порт:

```
mail:~/Desktop # rcripd start
Starting rip daemon (Zebra)
mail:~/Desktop # telnet 127.0.0.1 2602
done
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.

Hello, this is Quagga (version 0.99.15).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification

Password:
rip> show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
  (n) - normal, (s) - static, (d) - default, (r) - redistribute,
  (i) - interface

      Network           Next Hop           Metric From          Tag Time
C(i) 10.0.6.192/27    0.0.0.0           1 self               0
C(r) 192.168.40.0/24  0.0.0.0           1 self (connected:1) 0
rip>
```





На изображении представлена информация о двух сетях. Маршрутизатор имеет непосредственное подключение к каждой из этих сетей, об этом свидетельствует символ “C” в первом столбце вывода. При этом первая сеть, 10.0.6.192/27 – сеть, в которую RIP будет отсылать информацию для соседних маршрутизаторов, об этом свидетельствует символ “i” указанный в скобках после символа “C”. Вторая же сеть - 192.168.40.0/24, является «анонсируемой» (символ “r” в скобках после символа “C”), информация о данной сети будет распространяться текущим маршрутизатором для соседних роутеров.

После того, как текущий маршрутизатор получил информацию от соседнего роутера – он обновляет таблицу маршрутизации:

```
rip> show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
  (n) - normal, (s) - static, (d) - default, (r) - redistribute,
  (i) - interface

      Network          Next Hop          Metric From           Tag Time
C(i) 10.0.6.192/27    0.0.0.0            1 self              0
C(r) 192.168.40.0/24  0.0.0.0            1 self (connected:1) 0
R(n) 192.168.50.0/24  10.0.6.215         2 10.0.6.215        0 02:58
rip> ■
```

В таблице маршрутизации обнаружилась новая запись. Это информация о сети 192.168.50.0/24. Для того чтобы пакеты, в адресе которых указаны узлы из данной сети, достигли своего получателя, следует перенаправлять их на узел 10.0.6.215. Обратите на символ “R”, указанный в первом столбце последней строки – он свидетельствует о том, что данный маршрут был получен в результате работы протокола динамической маршрутизации RIP. Метрика маршрута свидетельствует о том, что между текущим маршрутизатором и целевой сетью присутствует еще один роутер.

Таким образом, динамическая маршрутизация нами успешно настроена.

### 23.3 Задания для самопроверки

1. Разбейте учебную сеть на три независимых сети. Объедините эти сети с помощью трех маршрутизаторов, подключенных в свою собственную четвертую сеть. Настройте статическую маршрутизацию. Проверьте результат.
2. Выполните предыдущее задание, используя динамическую маршрутизацию.



## 24. NFS

В информационной инфраструктуре современного предприятия задача создания распределенной или общей файловой системы является необходимой задачей. Использование NFS позволяет реализовать для пользователей прозрачную сеть. При соответствующей настройке, пользователи всегда будут использовать одинаковый профиль независимо от того, на каком рабочем месте они находятся.

Узел, который использует NFS, может одновременно выступать как в роли сервера, так и в роли клиента. Он может предоставлять ресурсы локальной файловой системы для других узлов сети (экспорт) и монтировать файловые системы, находящиеся на других узлах сети (импорт).

Для того чтобы использовать серверные функции NFS, следует установить пакет под названием *nfs-kernel-server*:

```
zypper install nfs-kernel-server
```

Для запуска NFS сервера (*nfsserver*) следует использовать следующий синтаксис:

```
/etc/init.d/nfsserver start
```

### 24.1 Импорт файловых систем с использованием NFS

Обязательным условием для успешного монтирования ресурсов NFS сервера – является запущенная служба *RPC port mapper*. Ее можно запустить следующей командой:

```
/etc/init.d/rpcbind start
```

После этого удаленная файловая система может быть смонтирована с помощью команды *mount* так, как будто это локальная файловая система:

```
mount узел:путь_к_удаленному_ресурсу локальный_путь
```

Например, для импорта домашних каталогов пользователей, которые находятся на узле *nfs.example.com*, используется следующая команда:

```
mount nfs.example.com:/home /home
```





Для автоматического монтирования удаленных файловых систем следует использовать службу *autofs*. Достаточно добавить следующую строчку в файл */etc/auto.master*:

```
/nfsmounts /etc/auto.nfs
```

Имя *auto.nfs* выбрано в соответствии с соглашением, но это может быть любое другое имя. В данном файле следует настраивать каждый удаленный ресурс, например:

```
Localdata -fstype=nfs server1:/data
nfs4mount -fstype=nfs4 server2:/
```

На примере каталогу */nfsmounts/Localdata* соответствует каталог */data*, расположенный на *server1*. Данный ресурс монтируется как NFS, в то время как ресурс */nfsmounts/nfs4mount*, расположенный на *server2* монтируется как NFSv4.

Монтировать удаленно расположенный NFS ресурс можно и с помощью файла */etc/fstab*. Типичная запись о NFSv3 ресурсе выглядит следующим образом:

```
nfs.example.com:/data /локальный/путь nfs rw,auto 0 0
```

Для монтирования NFSv4 ресурсов, следует использовать опцию *nfs4* вместо *nfs* в третьем столбце. После имени удаленного узла следует указывать символ псевдо корневого каталога ( “/” ):

```
nfs.example.com:/ /локальный/путь nfs4 rw,auto 0 0
```

Опция *auto* позволит смонтировать данные файловые системы при загрузке Linux. Это происходит по той причине, что в ходе загрузки выполняется команда *mount -a* в результате выполнения которой монтируются все системы с включенной опцией автоматического монтирования.

Если заменить опцию *auto* опцией *noauto*, автоматическое монтирование не будет производиться, но процесс ручного монтирования будет существенно упрощен, так как можно будет использовать довольно несложную команду. Например:

```
mount /локальный/путь
```



## 24.2 Экспорт файловых систем с использованием NFSv4

Для настройки службы NFS экспорта следует конфигурировать файл `/etc(exports` и, возможно, `/etc/sysconfig/nfs`. Для использования NFSv4, которая на текущий момент является самой актуальной версией данной службы, следует конфигурировать и файл `/etc/idmapd.conf`.

Файл `/etc(exports` содержит список, состоящий из нескольких строк. В каждой строке содержится описание того, какой каталог, и с какими параметрами предоставлен в общий доступ. Типичная строка файла `/etc(exports` выглядит так:

`/каталог/предоставляемый/в/общий/доступ узел (список_опций)`

Например:

```
/export 192.168.1.2(rw,fsid=0,sync,crossmnt)
/export/data 192.168.1.2(rw,bind=/data,sync)
```

В нашем примере узел `192.168.1.2` используется для идентификации клиентов, которым разрешен доступ к указанному ресурсу. В качестве клиентов можно указывать и DNS имена узлов, а также маски (например: `*.abc.com` или `*`).

Каталог, к которому указан параметр `fsid=0` – специальный. Он символизирует собой корень файловой системы, которую предоставляют в общий доступ. Такой каталог часто называют псевдо корневым. Этот каталог должен быть экспортирован также с опцией `crossmnt` для корректного взаимодействия с NFSv4. Все остальные каталоги должны экспортироваться с помощью NFSv4 начиная с указанной корневой точки. И если экспортируемый каталог не находится внутри псевдо корневого, следует «привязать» его к нему с помощью опции `bind`.

В частности, в рассматриваемом нами примере, каталог `/data` не находится внутри каталога `/export` на локальной файловой системе сервера. Поэтому мы экспортируем данный каталог с помощью каталога `/export/data` и осуществляем привязку данного каталога `/data` к данному имени.

При монтировании клиенты должны указывать лишь имя сервера `servername:/`, а не адрес вида `servername:/export`. Также нет необходимости в том, чтобы специально монтировать ресурс `servername:/data`. Он будет доступен для клиента автоматически внутри смонтированного псевдо корневого каталога.





Для поддержки экспорта с использованием NFSv4, следует отредактировать параметр *NFSv4\_SUPPORT*, в котором должно быть установлено значение “yes”.

Для успешной работы NFS сервера необходимо, чтобы имена UID пользователей на сервере и клиентов совпадали. Этого можно достичь с помощью копирования файла */etc/passwd* и */etc/shadow* или с использованием специальных служб каталогов: NIS или LDAP.

За сопоставление имен пользователей и их идентификаторов, отвечает служба *idmapd*, работа которой является необходимой для успешного функционирования NFSv4. Рассмотрим типичную конфигурацию данной службы, которая хранится в файле */etc/idmapd.conf*:

```
[General]
Verbosity = 0
Pipesfs-Directory = /var/lib/nfs/rpc_pipefs
Domain = Localdomain
[Mapping]
Nobody-User = nobody
Nobody-Group = nobody
```

Для перезапуска службы, после настройки ее конфигурационного файла, следует использовать следующую команду:

```
killall -HUP rpc.idmapd
```

А для успешного экспорта следует перезапустить службу *nfsserver*:

```
/etc/init.d/nfsserver restart
```

## 24.3 Экспорт с использованием NFSv2 и NFSv3

Экспорт с использованием NFSv2 и NFSv3 не имеет принципиальных отличий. Следует отредактировать параметры файлов */etc/exports* и */etc/sysconfig/nfs*. Каждая строка файла */etc/exports* соответствует следующему шаблону:

*/каталог/предоставленный/б/общий/доступ узел(список\_опций)*





например:

```
/export 192.168.1.2(rw,sync)
```

В данном примере каталог `/export` предоставлен в общий доступ для узла `192.168.1.2` со списком опций `rw, sync`. Этот IP адрес может быть заменен шаблоном вида `(*.*.abc.com)`.

После изменений в файлах `/etc/exports` или `/etc/sysconfig/nfs`, запустите или перезапустите службу `nfsserver`.

## 24.4 Задания для самопроверки

1. Настройте сеть между двумя узлами таким образом, чтобы пользователи, локально входящие на эти узлы, всегда имели доступ к одному и тому же содержимому их домашнего каталога.

