

27.1. Постановка задачи. Выбор конфигурации

Сначала определим функции, которые должен выполнять шлюз:

1. Поддержка связи с провайдером.
2. Маршрутизация IP-пакетов между локальной сетью и сетью Интернет для выхода пользователей локальной сети в Интернет.
3. Обеспечение IP-сервиса.
4. Защита локальной сети от несанкционированного доступа из Интернета.

Конфигурирование шлюза в операционной системе Linux состоит из следующих этапов:

1. Настройка ядра.
2. Настройка сети.
3. Конфигурирование IpTables.
4. Настройка DNS.
5. Настройка Squid.

Для определенности будет использоваться два сетевых интерфейса — `eth0`, идущий к провайдеру, и `eth1` — во внутренней сети. Пусть интерфейсу `eth0` назначен IP-адрес `111.111.111.111`, а `eth1` — `192.168.1.1`.

27.2. Пошаговое описание настройки шлюза

27.2.1. Настройка ядра

Скорее всего, вам придется перекомпилировать ядро. При этом должны быть активизированы следующие опции:

```
Networking support (CONFIG_NET) [y]
TCP/IP networking (CONFIG_INET) [y]
IP forwarding/gatewaying (CONFIG_IP_FORWARD) [y]
IP multicasting (CONFIG_IP_MULTICAST) [y]
IP firewalling (CONFIG_IP_FIREWALL) [y]
IP accounting (CONFIG_IP_ACCT) [y]
```

Можно также поэкспериментировать с набором опций **Advanced Router**, если данные функции есть в вашем ядре. Более подробно о процессе компилирования ядра вы можете прочитать в следующей главе.

27.2.2. Настройка сети

После перекомпилирования ядра нужно включить **IP-forwarding**. Сделайте это при помощи следующей команды:

```
# echo «1» > /proc/net/ip_forward
```

Настройку сетевых карт произведите с помощью программы **netconf**. О том, как это сделать, было рассказано в гл. 8.

27.2.3. Конфигурирование IPTables (или IpChains)

Теперь приступим к настройке **IPChains/IPTables**. В зависимости от вашего дистрибутива вы будете использовать один из этих бастаионов. Параллельно я буду писать команды **IPChains** и **IPTables**. Создайте цепочку, через которую пойдет весь трафик от провайдера:

<code>ipchains -N prov</code>	<code>iptables -N prov</code>
<code>ipchains -A input -i eth0 -j prov</code>	<code>iptables -A INPUT -i eth0 -j prov</code>

Запретите **ip-spoofing**:

<code>ipchains -A prov -s 192.168.1.1/16 -I -j DENY</code>	<code>iptables -A prov -s 192.168.1.1/16 -j DROP</code>
<code>ipchains -A prov -s 127.0.0.1/8 -I -j DENY</code>	<code>iptables -A prov -s 127.0.0.1/8 -I -j DROP</code>

Запретите **Telnet** снаружи:

```
ipchains -A prov -p tcp --destination-port 23 -j REJECT
```

Эта же команда «в переводе на **IPtables**» будет выглядеть так:

```
iptables -A prov -p tcp --destination-port 23 -j REJECT
```

Если вы не хотите, чтобы **samba** «светилась» наружу, запретите порты 137-139:

```
ipchains -A prov -p tcp --destination-port 137 -j REJECT
```

```
ipchains -A prov -p udp --destination-port 137 -j REJECT
```

То же самое сделайте для портов 138 и 129. На **IPTables** данные команды будут выглядеть так:

```
iptables -A prov -p tcp --destination-port 137 -j REJECT
```

```
iptables -A prov -p udp --destination-port 137 -j REJECT
```

О настройке **samba** вы можете прочитать в **Samba-HOWTO**.

Создайте цепочку для подсчета трафика:

```
ipchains -N trafiln
```

```
ipchains -I input -i eth0 -s ! 123.123.123.0/24 -p all -j trafiln
```

```
ipchains -A trafiln -d 123.123.123.123
```

Соответственно, на iptables команды будут выглядеть следующим образом:

```
iptables -N trafiln
iptables -I input -i eth0 -s ! 123.123.123.0/24 -p all -j trafiln
iptables -A trafiln -d 123.123.123.123
```

Для того, чтобы ваши правила были постоянными (при перезагрузке машины правила IpChains теряются), используйте скрипты **ipchains-save** и **ipchains-restore**. Настройте свои правила, а затем выполните команду:

```
# ipchains-save > /etc/ipchains.rules
# iptables-save > /etc/iptables.rules
```

Далее создайте скрипт, подобный тому, что приведен в листинге 27.1.

Листинг 27.1. Скрипт управления пакетной фильтрацией на ipchains

```
#!/bin/sh
# Скрипт управления пакетной фильтрацией.
# Если правил нет, то ничего не делать.
#!/bin/sh
# Скрипт управления пакетной фильтрацией.
# Если правил нет, то ничего не делать.
[ -f /etc/ipchains.rules ] || exit 0
case "$1" in
start)
echo -n "Включение пакетной фильтрации:"
/sbin/ipchains-restore < /etc/ipchains.rules || exit 1
echo 1 > /proc/sys/net/ipv4/ip_forward
echo "." ;;
stop)
echo -n "Отключение пакетной фильтрации:"
echo 0 > /proc/sys/net/ipv4/ip_forward
/sbin/ipchains -X
/sbin/ipchains -F
/sbin/ipchains -P input ACCEPT
/sbin/ipchains -P output ACCEPT
/sbin/ipchains -P forward ACCEPT
echo "." ;;
*)
echo "Использование: /etc/init.d/packetfilter {start|stop}"
exit 1 ;;
esac
exit 0
```

Этот скрипт добавьте в сценарии загрузки системы. Если вы используете IPTables, данный скрипт будет выглядеть так, как показано в листинге 27.2.

Листинг 27.2. Скрипт управления пакетной фильтрацией на IPTables

```

#!/bin/sh
# Скрипт управления пакетной фильтрацией.
# Если правил нет, то ничего не делать.
#!/bin/sh
# Скрипт управления пакетной фильтрацией.
# Если правил нет, то ничего не делать.
[ -f /etc/iptables.rules ] || exit 0
case "$1" in
start)
echo -n "Включение пакетной фильтрации:"
/sbin/iptables-restore < /etc/iptables.rules || exit 1
echo 1 > /proc/sys/net/ipv4/ip_forward
echo "." ;;
stop)
echo -n "Отключение пакетной фильтрации:"
echo 0 > /proc/sys/net/ipv4/ip_forward
/sbin/iptables -X
/sbin/iptables -F
/sbin/iptables -P INPUT ACCEPT
/sbin/iptables -P OUTPUT ACCEPT
/sbin/iptables -P FORWARD ACCEPT
echo "." ;;
*)
echo "Использование: /etc/init.d/packetfilter {start|stop}"
exit 1 ;;
esac
exit 0

```

27.2.4. Настройка DNS

Напомню, что основной задачей сервера доменных имен (Domain Name System) является преобразование мнемонических имен машин в IP-адреса и обратно. Обычно сервер DNS устанавливается на шлюзе, который используется для выхода в Интернет.

Прежде, чем приступить к настройке сервера, нужно определить, запущен ли он:

```
# ps -ax | grep named
```

Если он запущен, его нужно остановить (с помощью **ndc**), а если он вообще не установлен, то придется установить пакет **bind**. Для работы сервера должен быть активизирован сервис **network**.

Теперь приступим к непосредственной настройке сервера. Основная информация о параметрах сервера содержится в файле `/etc/named.conf` (см. листинг 27.3).

Листинг 27.3. Файл named.conf

```
logging {
    category cname {null; };
};
options {
    directory "/var/named";
};
zone "." {
    type hint;
    file "named.ca";
};
zone "dhsilabs.com" {
    type master;
    file "dhsilabs.com";
    notify no;
};
zone "0.0.127.in-addr.arpa" {
    type master;
    file "named.local";
};
zone "1.168.192.in-addr.arpa" {
    type master;
    file "192.168.1";
    notify yes;
};
```

Основной каталог сервера — /var/named. В нем сервер будет искать файлы dhsilabs.com, named.local, 192.168.1, named.ca (см. листинги 27.4, 27.5, 27.6). Обслуживаемая нашим сервером зона (домен) — dhsilabs.com (см. листинг 27.4). Файл named.ca — корневой кэш — содержит информацию о корневых серверах DNS. Позже займемся его обновлением.

Листинг 27.4. Файл dhsilabs.com
(для преобразования имен в IP-адреса)

```
@ IN SOA  den.dhsilabs.com. hostmaster.dhsilabs.com. (
    93011120  ; серийный номер
    10800    ; обновление каждые 3 часа
    3600     ; повтор каждый час
    3600000  ; хранить информацию 1000 часов
    86400)   ; TTL записи — 24 часа
IN NS den.dhsilabs.com.
IN A  192.168.1.1
IN MX 150  den.dhsilabs.com.
den  IN A  192.168.1.1
IN HINFOINTEL CELERON (LINUX)
IN MX 100  den
```

```
IN MX 150 evg.dhsilabs.com.
ns IN CNAME den.dhsilabs.com.
www IN CNAME den.dhsilabs.com.
ftp IN CNAME den.dhsilabs.com.
mail IN CNAME den.dhsilabs.com.
evg IN A 192.168.1.2
IN MX 100 den.dhsilabs.com.
localhost IN A 127.0.0.1
```

Здесь:

- ♦ NS — обозначает name server;
- ♦ A — IP-адрес;
- ♦ MX — почтовик <приоритет>. Чем ниже значение, тем выше приоритет;
- ♦ HINFO — сведения об аппаратном обеспечении (заполнять не рекомендуется);
- ♦ TXT — прочие сведения;
- ♦ CNAME — каноническое имя, т.е. если вы в окне браузера введете <http://www.dhsilabs.com>, то обращение будет произведено к den.dhsilabs.com.

Обратите внимание на точку в конце:

```
@ IN SOA den.dhsilabs.com. hostmaster.dhsilabs.com.
```

(Если точка не указана, то к имени будет добавлено имя домена (т.е. dhsilabs.com)).

Листинг 27.5. Файл named.local

```
@ IN SOA dhsilabs.com. root.dhsilabs.com. (
    199609203 ; серийный номер
    28800    ; обновление каждые 8 часов
    7200     ; повтор каждые 2 часа
    604800   ; хранить информацию 168 часов (7 дней)
    86400)   ; TTL записи — 24 часа
NS dhsilabs.com.
1 PTR localhost.
```

Листинг 27.6. Файл 192.168.1 или файл обратного соответствия

```
@ IN SOA den.dhsilabs.com. hostmaster.dhsilabs.com. (
    93011120 ; серийный номер
    10800    ; обновление каждые 3 часа
    3600     ; повтор каждый час
    3600000  ; хранить информацию 1000 часов
    86400)   ; TTL записи — 24 часа
```

```
@ IN NS den.dhsilabs.com
1 IN PTR den.dhsilabs.com
2.1.168.192 IN PTR evg.dhsilabs.com
```

Запись **PTR** используется для преобразования IP-адреса в имя. Если указан не весь IP:

```
1 IN PTR den.dhsilabs.com
```

то к нему будет добавлен адрес подсети 1.168.192. Обратите внимание: IP-адреса указываются в обратном порядке!

27.2.5. Настройка прокси Squid

Установите пакет **squid**. Осталось настроить и запустить его. Для этого нужно отредактировать файл конфигурации `/etc/squid/squid.conf`. Сначала укажите адрес прокси-провайдера:

```
cache_peer proxy.your_isp.com
```

Задайте объем ОЗУ, который будет использовать прокси-сервером:

```
cache_mem
```

В том случае, если вы планируете использовать этот компьютер еще и для других целей, кроме как в качестве прокси-сервера, то не рекомендуется устанавливать здесь более трети физического объема ОЗУ.

Далее укажите, где будет располагаться кэш (первое число — это количество Мб для кэша):

```
cache_dir /usr/local/squid 2048 16 256
```

Затем укажите хосты, из которых разрешен доступ к прокси-серверу:

```
acl allowed_hosts src 192.168.1.0/255.255.255.0
acl localhost src 127.0.0.1/255.255.255.255
```

После этого пропишите пользователей, которым разрешено пользоваться **squid** (в приведенном примере это `den`, `admin`, `developer`):

```
ident_lookup on
acl allowed_users user den admin developer
http_access allow allowed_users
http_access deny all
```

Тэги **maximum_object_size** и **maximum_object** устанавливают ограничения на размер передаваемых объектов.

В заключение хочу дать один хороший совет: из соображений безопасности отредактируйте свои `/etc/services` и `/etc/inetd.conf` и отключите неиспользуемые сервисы — это уменьшит вероятность взлома вашей системы. Вот, в общем-то, и все.

Настройка Dial-In сервера (сервера доступа)

28.1. Простейшее решение

28.1.1. Установка необходимого программного обеспечения

В качестве операционной системы, естественно, будем использовать ОС Linux. Метод настройки, рассмотренный в этой главе, подойдет для любого дистрибутива. Также вам потребуются пакет **ppp** версии не младше 2.3.x (желательно самая новая версия) и пакет **mgetty**. Пакет **mgetty** должен быть собран с опцией **-DAUTO_PPP**. Если это не так, то его нужно пересобрать.

Если вы используете RedHat/Mandrake, установить **ppp** и **mgetty** можно с помощью команд:

```
# mount -t iso9660 /dev/hdd /mnt/cdrom
#cd /mnt/cdrom/Mandrake/RPMS
#rpm -Uvh mgetty*
#rpm -Uvh ppp*
```

Некоторые замечания:

1. CD-ROM является устройством `/dev/hdd` (или `Secondary Slave`).
2. Используется Linux Mandrake. При использовании Red Hat пакеты находятся в `/mnt/cdrom/RedHat/RPMS`.
3. Не используется **supermount**. Если у вас **supermount** активен, первую команду вводить не нужно.
4. Третья и четвертая команды устанавливают все семейство **mgetty** и **ppp**. При использовании такого подхода устанавливаются все файлы — и никакой заботы! Но при этом вы можете установить и только то, что вам нужно.

28.1.2. Настройка mgetty

При корректной сборке или установке пакетов **mgetty** и **ppp** у вас должны быть следующие файлы:

```
Каталог /etc/mgetty+sendfax:
dialin.config
login.config
mgetty.config
Каталог /etc/ppp:
auth-up
auth-down
chap-secrets
ip-up
ip-down
options
pap-secrets
```

Если их нет, вам нужно самостоятельно найти, где они находятся. При самосборке смотрите, что и куда проинсталлировалось. В крайнем случае необходимые файлы придется создать вручную.

Файл `/etc/mgetty+sendfax/dialin.config` обычно пустой — в нем все закомментировано. Файл `/etc/mgetty+sendfax/login.config` должен содержать строчку:

```
/AutoPPP/- a_ppp /etc/ppp/ppplogin
```

Убедитесь, что эта строчка не закомментирована. Если вы хотите, чтобы имена пользователей записывались в журналы (log-файлы), отредактируйте эту строку следующим образом:

```
/AutoPPP/- - /etc/ppp/ppplogin
```

Затем создайте файл `/etc/ppp/ppplogin` (см. листинг 28.1).

Листинг 28.1. Файл `/etc/ppp/ppplogin`

```
mesg n
tty -echo
/usr/sbin/pppd silent auth -chap +pap login
```

В некоторых версиях **ppp** вместо **-chap** нужно писать **refuse-chap**, а вместо **+pap** писать **require-pap**. Продолжая настройку, сделайте `/etc/ppp/ppplogin` исполняемым:

```
# chmod +x /etc/ppp/ppplogin
```

В приведенном примере используется PAP-аутентификация с использованием пароля из файла `/etc/passwd` (см. ниже). Файл `/etc/mgetty+sendfax/mgetty.config` должен быть примерно такой, как в листинге 28.2.

Листинг 28.2. Файл mgetty.config

```
# For US Robotics Sportster 28.8 with speaker off
port ttyS0
speed 28800
data-only y
debug 3
init-chat "" ATZ OK AT&F1M0E1Q0S0=0 OK
answer-chat "" ATA CONNECT \c \r
# For Practical Peripheral 14.4 with fax disabled and prolonged
# carrier wait time (90 sec)
port ttyS1
speed 14400
data-only y
debug 3
init-chat "" ATZ OK AT&F1M0E1Q0S0=0S7=90+FCLASS=0 OK
answer-chat "" ATA CONNECT \c \r
# For USRobotics V.Everything
port ttyS2
speed 57600
data-only y
debug 3
init-chat "" AT OK ATS7=50S0=1+S62=3+S64=2S39=10 OK
Для ZyXEL U336E можно использовать такие параметры:
init-chat "" ATZ OK AT&F1M0E1Q0S0=0S OK
answer-chat "" ATA CONNECT \c \r
```

В нем определены параметры для трех модемов: US Robotics Sportster 28.8, Practical Peripheral 14.4, USRobotics V.Everything. Для ZyXEL U336E можно использовать такие параметры:

```
init-chat "" ATZ OK AT&F1M0E1Q0S0=0S OK
answer-chat "" ATA CONNECT \c \r
```

Данные параметры вы можете узнать из документации по вашему модему. Очень рекомендую прочитать ее перед установкой сервера входящих звонков.

Теперь нужно изменить файл /etc/inittab как это показано в листинге 28.3.

Листинг 28.3. Фрагмент файла inittab

```
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
```

```
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
#эти строки нужно добавить
S0:2345:respawn:/sbin/mgetty -x 3 ttyS0
S1:2345:respawn:/sbin/mgetty -x 3 ttyS1
S2:2345:respawn:/sbin/mgetty /dev/ttyS2
```

Здесь **S0**, **S1**, **S2** — это просто идентификаторы. Вы можете использовать вместо них любое имя. Нужно только назначить отдельное имя для каждого порта. Имена **S0...S2** я использовал для наглядности.

Теперь запустите **mgetty** (перед выполнением этой команды следует включить модемы):

```
# init q
```

Если при выполнении этой команды модемы не подключены или выключены, в файле `/var/log/messages` вы получите сообщение об этом. Если на модеме загорелась лампочка **TR**, то все настройки выполнены правильно и **mgetty** «подхватил» модем.

28.1.3. Настройка ppp

Обычно для каждого порта в каталоге `/etc/ppp` создается файл `options.ttySx`, где **x** — номер порта (см. листинг 28.4).

Листинг 28.4. Файл options

```
lock
login
auth
netmask 255.255.255.0
modem
crttscts
refuse-chap
require-pap
mtu 576
mru 576
proxyarp
myhost:ppp01
ms-dns CCC.CCC.CCC.CCC
```

Общие настройки для всех портов можно вынести в файл `/etc/ppp/options`. Имя `myhost` замените на реальное имя вашего сервера входящих звонков.

Обратите внимание, что **ppp01** — это произвольно выбранное имя виртуального узла абонента. Вы можете использовать другие имена, например, `igor`, `denis` и тому подобное.

Имена узлов должны быть уникальными, т.е. если вы используете **ppp0** в `options.ttyS0`, то в `options.ttyS1` нужно использовать **ppp01** и так далее.

Опция **ms-dns** определяет сервер DNS для клиентов Microsoft. Укажите здесь IP-адрес сервера DNS вашей сети.

Используйте опцию **proxyarp**, так как IP-адреса будут назначаться внутри **broadcast** ваших сетевых карт локальной сети. При этом демон **pppd** будет делать вид, что виртуальный хост находится внутри вашего сегмента ethernet. Небольшая деталь: вместо опции **refuse-chap** можно писать **-chap**, а вместо **require-pap** писать **+pap**.

Теперь отредактируйте файл `/etc/ppp/pap-secrets` (см. листинг 28.5.)

Листинг 28.5. Файл `/etc/ppp/pap-secrets`

```
# Secrets for authentication using PAP
# client  server  secret  IP addresses
* * " " 192.168.0.11
* * " " 192.168.0.12
* * " " 192.168.0.13
```

В приведенном примере используются три модема для входящих звонков, поэтому нужно сделать три записи. Пароли пользователей находятся в файле `/etc/passwd` (или `/etc/shadow`). Соответственно внесите изменения в свой `/etc/hosts` (см. листинг 28.6)

Листинг 28.6. Файл `/etc/hosts`

```
192.168.0.11 ppp01 ppp01.mydomain.com
192.168.0.12 ppp02 ppp02.mydomain.com
192.168.0.13 ppp03 ppp03.mydomain.com
```

Имя `mydomain.com` замените на реальное имя домена. По большому счету эти записи не мешало бы также внести и в локальную зону DNS. Теперь осталось установить нужные права доступа для `/usr/sbin/pppd`:

```
# chmod u+s /usr/sbin/pppd
```

28.1.4. Включение IP Forwarding

Разрешение пересылки IP устанавливается в файле `/etc/sysconfig/network` примерно так: `FORWARD_IPV4=yes`. При этом ваше ядро должно быть скомпилировано для поддержки `IP_FORWARD`.

Для включения IP Forwarding введите команду:

```
# echo «1» > /proc/net/ip_forward
```

В некоторых дистрибутивах IP Forwarding включается несколько иначе. Если в вашем дистрибутиве есть программа **netconf**, используйте ее для включения IP Forwarding, если ее нет, изучите документацию по вашему дистрибутиву.

Теперь вы готовы к работе!

28.1.5. Альтернативный вариант настройки

Этот вариант может оказаться даже более простым, чем первый. Настройки файлов `/etc/options` и `/etc/options.ttySx` остаются прежними, но строку **myhost:pp01** нужно заменить на строку вида:

```
Server_IP:Client_IP
```

Например,

```
192.168.0.1:192.168.0.11
```

Теперь нужно изменить содержание файла `/etc/ppp/pap-secrets` (см. листинг 28.7).

Листинг 28.7. Фрагмент файла `/etc/ppp/pap-secrets`

```
#
user1 сервер.домен "" *
user2 сервер.домен "" *
#
```

где: **user1** — это имя пользователя, зарегистрированного в системе;

сервер.домен — это имя сервера входящих звонков;

«» — пароли брать из `/etc/passwd` (`/etc/shadow`);

***** — абонент может аутентифицироваться с любого IP.

При желании можно назначить другой пароль. В этом случае, если данный сервер используется также и в качестве почтовика, то для входящих звонков и сервиса POP будут использоваться различные пароли.

Внимание! Пароли в `/etc/ppp/pap-secrets` содержатся в открытом виде и не кодируются с помощью алгоритма MD5 (или DES), как в файле `/etc/shadow` (`/etc/passwd`). Файл `/etc/hosts` править не нужно.

Вот, собственно, и все.

28.1.6. Если что-то не работает...

Лучший совет в этом случае — смотрите файл `/var/log/messages`. В нем много всего интересного. Если у вас появляются сообщения вида:

```
modprobe: can't locate module char-major-24
```

то надо прописать в файле `/etc/conf.modules` следующие строки:

```
alias ppp-compress-21 bsd_comp
alias ppp-compress-24 ppp_deflate
alias ppp-compress-26 ppp_deflate
```

28.1.7. Настройка Windows-клиентов

Рассмотрим наиболее распространенную ситуацию, когда для подключения к нашему серверу используется обыкновенное удаленное соединение. IP-адрес клиента и адрес сервера DNS назначается провайдером, то есть нашим сервером. При этом в свойствах соединения нужно указать следующие данные (см. рис. 28.1):

- ♦ Тип сервера удаленного доступа: PPP.
- ♦ Дополнительные параметры: только «Программное сжатие данных».
- ♦ Допустимые протоколы: только «TCP/IP».

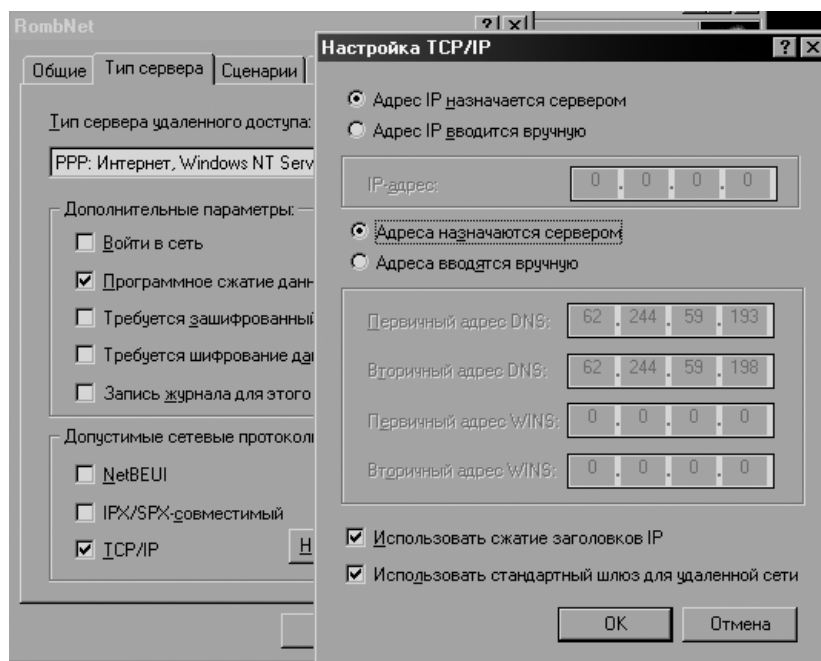


Рис. 28.1. Параметры PPP

28.2. Настройка Dial-In сервера с использованием протокола RADIUS

28.2.1. Установка FreeRADIUS

Описание и возможности протокола RADIUS

RADIUS (Remote Authentication Dial In User Service) — это протокол и программное обеспечение, позволяющее серверам удаленного доступа (RAS) «общаться» с центральным сервером, аутентифицируя пользователей и предоставляя им доступ к сетевым сервисам. Попросту говоря, RADIUS можно использовать для учета времени работы пользователей. В п. 28.1 этой главы мы настроили сервер удаленного доступа — Dial In (или RAS), который предоставляет доступ пользователей к сети Интернет (и к нашей локальной сети), но ведь нужно же знать, сколько «присидел в Интернете» конкретный пользователь.

Как же работает RADIUS? Когда пользователь дозвонился на RAS, происходит следующее:

- Начинается процедура аутентификации пользователя. В связи с этим RAS пытается установить соединение с первым RADIUS-сервером.
- Если соединение не установлено (например, сервер загружен или вообще выключен), RAS или запрашивает данный сервер еще раз, или пытается подключиться к следующему серверу.
- Если соединение установлено, RADIUS-сервер проверяет IP-адрес и ключ RAS (сначала RADIUS проверяет сервер удаленного доступа, а потом уже пользователя). Если IP-адрес и ключ правильные, RADIUS приступает к проверке клиента. В противном случае клиенту посылается сообщение Invalid Key.
- Проверка «подлинности» клиента заключается в проверке его пароля (ясное дело, по сети передается не сам пароль, а его MD5-хеш). Можно также настроить RADIUS для проверки IP-адреса клиента и порта RAS. В случае, если клиент не прошел аутентификацию, посылается сообщение Access denied с указанием кода (иногда и описания) ошибки.
- Если же пароль правильный, пользователю разрешается доступ. Точнее, пользователю посылается пакет, содержащий информацию о том, что доступ разрешен, и о специфических параметрах соединения: протокол (PPP, SLIP), MTU, IP-адрес и др.

В сети имеется очень много серверов RADIUS — FreeRadius, XtRadius, Cistron RADIUS, Gnu-Radius, IC-RADIUS, Radiusd-livingston. Все они обладают своими преимуществами и недостатками. Например, Radiusd-livingston очень прост в настройке, но не очень удобен в эксплуатации, потому что не поддерживает SQL-серверы. А это не очень удобно, если у вас несколько десятков клиентов: на практике количество клиентов

провайдера значительно превышает 4-5 десятков, у хорошего провайдера оно исчисляется в тысячах. FreeRadius сложнее в конфигурировании, зато намного проще в эксплуатации: он поддерживает SQL-серверы (mysql, postgresql, oracle, LDAP), имеет Web-интерфейс и поддерживает дополнительные модули.

Я остановил свой выбор на FreeRadius, который будет описан в этой книге. Сервер FreeRadius абсолютно бесплатно доступен по адресу <http://www.freeradius.org> (файл <ftp://ftp.freeradius.org/pub/radius/freeradius.tar.gz> — 1775 Кб). Последняя версия — 1.0.2.

Устанавливаем FreeRADIUS

Компиляция сервера обычно не вызывает проблем. Просто распакуйте содержимое архива `freeradius.tar.gz` в каталог `/usr/src/freeradius`, а затем выполните команды:

```
./configure
make
make install
```

После установки FreeRadius в каталоге `/etc/raddb` появятся следующие файлы:

- ♦ `radiusd.conf` — основной файл конфигурации FreeRadius;
- ♦ `clients.conf` — настройка клиентов (имеются в виду серверы удаленного доступа — RAS, а не клиенты провайдера);
- ♦ `proxy.conf` — конфигурация прокси-сервера. Данный файл может использоваться для перенаправления запросов клиентов, то есть роуминга клиентов. Об этом файле мы поговорим чуть позже;
- ♦ `users` — описывает установки пользователей (это уже клиенты провайдера);
- ♦ `acct_users` — установки учета пользователей;
- ♦ `sql.conf` — установки сервера MySQL;
- ♦ `attrs` — информация об AV-парах по умолчанию для каждой области, определенной в файле `proxy.conf`;
- ♦ `dictionary.conf` — информация обо всех известных AV-парах;
- ♦ `hints` — информация о суффиксах и префиксах имен пользователей.



Примечание

AV-парой называется пара атрибут=значение, например, `Auth-Type=System`. Почему AV-пары называются именно AV-парами? Атрибут переводится на английский как `attribute`, а значение — как `value`, вот и получается `attribute = value`, AV-пара.

28.2.2. Настройка FreeRADIUS. Редактирование файлов конфигурации

Основные настройки. Файл radiusd.conf

Начнем с основного файла конфигурации — radiusd.conf. Данный файл очень объемный, поэтому рассматривать его будем не полностью, а частями. Рассмотрим некоторые полезные опции этого файла (см. листинг 28.8).

Листинг 28.8. Фрагмент файла radiusd.conf

```
# Файл конфигурации прокси
# $INCLUDE ${confdir}/proxy.conf
# Файл конфигурации серверов RAS
$INCLUDE ${confdir}/clients.conf
# Из соображений безопасности будем запускать сервер от имени
# пользователя nobody и группы nogroup
user = nobody
group = nogroup
# Максимальное время обработки запроса (в секундах),
# по окончании которого будет разорвано соединение с клиентом
max_request_time = 30
# Максимальное число клиентов для данного RADIUS-сервера.
# Для получения этого числа нужно умножить количество
# RAS серверов на 256. У нас RAS-сервер один-единственный,
# поэтому максимальное число клиентов равно 256
max_requests = 256
# Протоколировать ли данные о полном имени пользователя
log_stripped_names = yes
# Протоколировать ли информацию о запросах на аутентификацию
log_auth = no
# Протоколировать ли "правильные" и "неправильные" пароли
log_auth_goodpass = no
log_auth_badpass = yes
# Приводить ли к нижнему регистру имя пользователя и пароль.
# Возможны значения:
# no - проверять имя пользователя и пароль как есть
# before- сначала привести к нижнему регистру, а потом аутентифицировать
# after - проверить как есть, если пароль или имя пользователя
# неправильные, привести к нижнему регистру и проверить еще раз
lower_user = no
lower_pass = no
```

```
# Параметры безопасности
security {
# Максимальное число AV-пар в пакете
max_attributes = 100
# Пакет отказа в доступе будет посылаться с определенной
# задержкой. Задержка позволяет
# предотвратить DOS-атаку (атака "отказ в обслуживании")
reject_delay = 2
# Не посылать ответ на запрос RAS о статусе
status_server = no
}
# Мы не используем перенаправление, поэтому директива
# включения файла proxy.conf закомментирована (см. выше).
# Если вам нужно перенаправление (об этом мы поговорим),
# раскомментируйте эту строку (в начале файла)
#
proxy_requests = no
```

Теперь перейдем к изменению параметров серверов RAS (см. листинг 28.9).

Листинг 28.9. Файл clients.conf

```
# Определяем параметры RAS-сервером.
# Формат 'client [имя_узла|ip-адрес]'
#
client 127.0.0.1 {
# Пароль для идентификации клиента. Максимальная длина пароля –
# 32 символа. Измените пароль по умолчанию!
secret      = testing123
# Короткое имя клиента
shortname = localhost
# Тип RAS-сервера
# Допустимые типы:
#
# cisco
# computone
# livingston
# max40xx
# multitech
# netserver
# pathras
# patton
```

```
# portslave
# tc
# usrhiper
# other # все другие типы
nastype = other # localhost обычно не является RAS
# Две следующие строки зарезервированы на будущее, то есть
# пока не используются и их изменения не приведет к каким-либо
# последствиям
# login= !root
# password = someadminpas
}
# Описание клиентов. У нас он один
client 192.168.1.2 {
secret      = testing123
shortname   = localhost
}
# Можно определить целую сеть клиентов — RAS-серверов
#client 192.168.0.0/24 {
# secret      = testing123-1
# shortname   = private-network-1
#}
```

Файл `проху.conf` — задаем распределение нагрузки

Следующий файл — `проху.conf`. Как вы уже знаете, этот файл используется для распределения нагрузки на сервер — роуминга пользователей. Например, вы можете заключить договор с другим провайдером, согласно которому он будет обслуживать некоторых ваших клиентов, а вы — его клиентов. Такое перенаправление целесообразно, когда у вас *очень* много клиентов и ваши серверы не справляются с обработкой запросов. Вряд ли вы будете использовать этот файл, но знать о нем нужно. Данный файл довольно объемный, мы рассмотрим только те параметры, которые касаются перенаправления пользователей (см. листинг 28.10), остальные параметры вполне допустимы по умолчанию.

Листинг 28.10. Фрагмент файла `проху.conf`

```
# Область isp2.com (домен isp2.com)
realm isp2.com {
# тип сервера
type = radius
# сервер для аутентификации
authhost = radius1.isp2.com:1645
```

```
# сервер для учета
accthost = radius1.isp2.com:1646
# это наш секрет :-)
secret = TheirKey1
# если опция nostrip не указана, то запрос от пользователя
# "user@company.com" будет отправлен
# серверу radius1.isp2.com как "user". Если опция nostrip,
# запрос будет отправлен как
# "user@company.com"
nostrip
}
# Если перенаправить запрос на сервер radius1 не удалось,
# RADIUS-сервер (наш) будет искать
# другой сервер. Опишем его:
realm isp2.com {
    type = radius
    authhost = radius2.isp2.com:1645
    accthost = radius2.isp2.com:1646
    secret = TheirKey2
}
# Можно описать область NULL для пользователей,
# которые не определили свою область
realm NULL {
    type = radius
    authhost = radius2.company.com:1600
    accthost = radius2.company.com:1601
    secret = TheirKey
}
# Всех остальных пользователей перенаправим в область DEFAULT
realm DEFAULT {
    type = radius
    authhost = radius.company2.com:1600
    accthost = radius.company2.com:1601
    secret = key1231
}
```

Описываем клиентов. Файл users

Разобравшись с перенаправлением клиентов, опишем теперь самих клиентов. Установки клиентов описываются в файле `users`. Если у вас большое число клиентов, использовать данный файл не очень удобно. На практике обычно используется сервер баз данных. В этой книге мы будем использовать сервер баз данных MySQL, но файл `clients` все же опишем для полноты обзора (см. листинг 28.11).

Листинг 28.11. Установки пользователей — файл users

```
denisAuth-Type := Local, User-Password == "testing"
Service-Type = Framed-User,
Framed-Protocol = PPP,
Framed-IP-Address = 192.168.1.100,
Framed-IP-Netmask = 255.255.255.0,
Framed-Routing = Broadcast-Listen,
Framed-MTU = 1500,
Framed-Compression = Van-Jacobson-TCP-IP
test Auth-Type := Accept, User-Password == "testing"
Service-Type = Framed-User,
Framed-Protocol = PPP,
Framed-IP-Address = 192.168.1.101,
Framed-IP-Netmask = 255.255.255.0,
Framed-Routing = Broadcast-Listen,
Framed-MTU = 1500,
Framed-Compression = Van-Jacobson-TCP-IP
lamerAuth-Type := Reject
Reply-Message = "Your account has been disabled."
```

Я советую разобраться с этим файлом подробнее, поскольку в нем описываются параметры, которые позже мы будем вносить в таблицы MySQL — там я не буду подробно останавливаться на типах аутентификации, MTU и прочих параметрах.

Начнем с самого первого пользователя — `denis`. Используется тип аутентификации `Local`. Это означает, что пароль будет браться из файла `users`. Я рекомендую использовать тип аутентификации `System`. В этом случае пароль для сравнения будет определяться из файлов `/etc/passwd` и `/etc/shadow` — там он хоть недоступен для «невооруженного взгляда». Аутентификация `System` используется для PAP-протокола.

Полезными окажутся следующие типы аутентификации:

- ♦ **Accept** — сразу возвращает пакет разрешения доступа. Используется для тестирования соединения или так называемого тестового входа, позволяющего будущим клиентам оценить качество предоставляемых услуг.
- ♦ **Reject** — сразу возвращает пакет запрещения доступа. Можно использовать для блокирования пользователей, которые давно не пользовались услугами провайдера, хотя для этих целей существует AV-пара `Expiration`, например, `Expiration = «Jan 01 2004»`.
- ♦ **Chap** — используется протокол Chap (пароли передаются в открытом виде!).
- ♦ **Ms-Chap Ldap** — для проверки пароля используется служба каталогов `ldap`. Этот способ требует особой настройки как `FreeRADIUS`, так и службы каталогов, поэтому мы его не будем рассматривать.

AV-пару User-Password нужно указывать, если вы планируете использовать тип аутентификации Local, что я не рекомендую делать. Вообще по возможности старайтесь нигде не прописывать пароли пользователей в открытом виде.

В AV-парах можно использовать операторы, перечисленные в табл. 28.1. Напомню, что все AV-пары указываются в формате:

Attribute operator Value

Операторы, используемые в AV-парах

Таблица 28.1

Оператор	Действие
=	Атрибут будет отправлен RAS, значение его равно Value
:=	Атрибут принимает указанное значение, если он до этого не был определен
==	Используется для проверки значения Value
+=	Добавление нового значения к атрибуту
!= (не равно), <, >, <=, >=	Операторы сравнения
=*	Проверяет, содержит ли запрос данный атрибут, значение роли не играет
=~	Проверяет соответствие значения атрибута регулярному выражению

Параметры учета пользователей. Файл acct_users

Файл acct_users содержит параметры учета пользователей. Пример этого файла приведен в листинге 28.12.

Листинг 28.12. Файл acct_users

```
# Данная программа будет запущена при начале сессии
DEFAULT Acct-Status-Type == Start
Exec-Program = "/path/to/exec/acct/start %U"
# А эта — по окончании сессии
DEFAULT Acct-Status-Type == Stop
Exec-Program = "/path/to/exec/acct/stop %U"
```

Программы **start** и **stop** являются счетчиками. Что они считают — зависит от реализации этих программ: можно считать время, а можно — трафик.

В файле acct_users можно использовать специальные переменные, например, %U — это имя пользователя. Список всех переменных вы найдете в документации по FreeRADIUS. Вот наиболее полезные, на мой взгляд, переменные:

```
%U ..... имя пользователя (user);
%u ..... полное имя пользователя (user@company.com);
```

%a протокол (PPP или SLIP);
%d, %m, %Y .. день, месяц, год;
%n IP-адрес сервера RAS (в терминологии FreeRADIUS серверы RAS называются NAS (Network Access Server) — вот отсюда буква «n»);
%C имя клиента;
%S время и дата в формате SQL;
%M MTU;
%p номер порта.

Напомню, что файл `acct_users` используется для учета времени работы без использования MySQL (в паре с файлом `users`).

28.2.3. FreeRADIUS и MySQL

Вот теперь можно смело перейти к настройке MySQL. Надеюсь, он уже у вас установлен. Если нет, рекомендую прочитать соответствующую главу этой книги. Если MySQL настроен и запущен, остановите его:

```
service mysqld stop
```

После этого установите пакет **mysql-dev**.



Примечание

Если этот пакет установлен, останавливать сервер не нужно

Теперь запустите MySQL и создайте базу данных `radius`:

```
mysql -u root -p
mysql> create database radius
mysql> quit
```

Перейдите в каталог, в который вы распаковали архив `freeradius`. Затем выполните команду:

```
cd ./usr/modules/rlm_sql/drivers/rlm_sql_mysql/
```

Создаем системные таблицы, необходимые для FreeRADIUS:

```
mysql -uroot -pПароль radius < db_mysql.sql
```

Сценарий **db_mysql.sql** создаст шесть таблиц:

- ♦ `radacct` — основная таблица, используемая для учета пользователей;
- ♦ `radcheck` — проверка пользователей;
- ♦ `usergroup` — группы пользователей;
- ♦ `radcheckgroup` — проверка групп пользователей;

- ♦ `radreply` — содержит AV-пары, которые будут передаваться клиенту при успешной аутентификации;
- ♦ `radgroupreply` — ответ для группы.

Основными таблицами являются `radacct`, `radcheck` и `radreply`. Первую таблицу вы не будете редактировать вручную, а вот вторые две придется отредактировать.

В таблицу `radcheck` нужно добавить пользователя и две AV-пары, задающие тип аутентификации и пароль пользователя, если используется тип аутентификации `Local`.

```
mysql> select * from radcheck;
```

id	UserName	Attribute	op	Value
1	denis	Auth-Type	:=	Local
2	denis	Password	==	testing
3	dhsilabs	Auth-Type	:=	System

Добавить нового пользователя позволяет следующий SQL-запрос:

```
INSERT into radcheck VALUES('', 'max', 'Auth-Type', ':=', 'System');
```

Если вы используете аутентификацию `Local`, вам нужно выполнить два запроса:

```
INSERT into radcheck VALUES('', 'max', 'Auth-Type', ':=', 'Local');
INSERT into radcheck VALUES('', 'max', 'Password', '==', '123456');
```

Для изменения параметров уже добавленного пользователя нужно использовать SQL-оператор `UPDATE`. Подробнее об этом операторе читайте в гл. 17.

В таблице `radreply` содержатся AV-пары, которые будут переданы клиенту, если он успешно прошел аутентификацию.

```
mysql> select * from radperly;
```

Id	UserName	Attribute	Op	Value
1	denis	Framed-IP-Address	=	192.168.1.100
2	denis	Framed-IP-Netmask	=	255.255.255.0
3	denis	Framed-MTU	=	1500
4	denis	Framed-Protocol	=	PPP

Мы уже добавили нескольких пользователей, определили их параметры, но до сих пор не «прикрутили» MySQL к FreeRADIUS. Откройте файл `radiusd.conf` и найдите секцию `modules`. Убедитесь, что MySQL включен (подключен) файл `sql.conf`:

```
modules {  
...  
$INCLUDE ${confdir}/sq.conf  
}
```

Затем настроим секцию авторизации. Она должна выглядеть следующим образом:

```
authorize {  
preprocess  
# используем PAP. Если вы используете chap, раскомментируйте  
# строку ниже, а эту нужно закомментировать  
pap  
# chap  
suffix  
# для авторизации использовать файл users или MySQL.  
# Мы используем MySQL files  
sql  
}
```

Теперь открываем файл `sql.conf` и редактируем его (см. листинг 28.13).

Листинг 28.13. Фрагмент файла настроек MySQL – `sql.conf`

```
sql {  
# Тип сервер баз данных  
# Сейчас поддерживаются серверы: rlm_sql_mysql, rlm_sql_postgresql,  
# rlm_sql_iodbc, rlm_sql_oracle, rlm_sql_unixodbc, rlm_sql_freetds  
driver = "rlm_sql_mysql"  
# Информация о соединении  
server = "localhost"  
login = "root"  
password = "rootpass"  
# Название базы данных  
radius_db = "radius"  
# Прописываем наши таблицы  
acct_table1 = "radacct"  
acct_table2 = "radacct"  
  
authcheck_table = "radcheck"  
authreply_table = "radreply"  
groupcheck_table = "radgroupcheck"  
groupreply_table = "radgroupreply"  
usergroup_table = "usergroup"  
...  
}
```

Сейчас мы вплотную подошли к аккаунтингу, то есть учету времени работы пользователей. Найдите секцию accounting файле `radiusd.conf`:

```
accounting {  
    # Данный модуль создает уникальный идентификатор пакета аккаунтинга  
    # Требуется большинством RAS  
    acct_unique  
    # Модуль detail позволяет создавать каждый час файлы, содержащие  
    # информацию о работе пользователя. Файлы создаются в каталоге  
    # ${radacctdit}/%Client-IP-Address/detail-$Y%m%d  
    # detail  
    # Аккаунтинг через SQL  
    sql  
}
```

28.2.4. Учет времени работы пользователей

Нам осталось написать счетчик, подсчитывающий время работы пользователя. Возьмем готовый счетчик, размещенный на сайте www.freeradius.org. Файл называется `rlm_sqlcounter` (http://www.freeradius.org/radiusd/doc/rlm_sqlcounter). Переименуем его в `sqlcounter.conf` (файл нужно поместить в каталог `/etc/raddb`). Текст файла приведен в листинге 28.14.

Листинг 28.14. Счетчик учета времени работы пользователя

```
# Счетчик использует таблицу radacct  
sqlcounter noresetcounter {  
    counter-name = Max-All-Session-Time  
    check-name = Max-All-Session  
    sqlmod-inst = sql  
    key = User-Name  
    reset = never  
    query = "SELECT SUM(AcctSessionTime) FROM radacct WHERE  
    UserName='%{k}' "  
}  
sqlcounter dailycounter {  
    driver = "rlm_sqlcounter"  
    counter-name = Daily-Session-Time  
    check-name = Max-Daily-Session  
    sqlmod-inst = sql  
    key = User-Name  
    reset = daily
```

Часть VI. Практические примеры полной настройки Linux-сервера

```
        query = "SELECT SUM(AcctSessionTime - GREATEST((%b -
        UNIX_TIMESTAMP(AcctStartTime)), 0)) FROM radacct WHERE
        UserName='%{k}' AND UNIX_TIMESTAMP(AcctStartTime) +
        AcctSessionTime > '%b'"
    }
    sqlcounter monthlycounter {
        counter-name = Monthly-Session-Time
        check-name = Max-Monthly-Session
        sqlmod-inst = sql
        key = User-Name
        reset = monthly
        query = "SELECT SUM(AcctSessionTime - GREATEST((%b -
        UNIX_TIMESTAMP(AcctStartTime)), 0)) FROM radacct WHERE
        UserName='%{k}' AND UNIX_TIMESTAMP(AcctStartTime) +
        AcctSessionTime > '%b'"
    }
}
```

Пропишем счетчик в файле конфигурации FreeRADIUS — radiusd.conf:

```
modules {
...
$INCLUDE ${confdir}/sqlcounter.conf
...
}
```

Затем подключаем счетчик к процедуре авторизации:

```
authorize {
...
noresetcounter
dailycounter
monthlycounter
}
```

После добавления счетчика в секцию авторизации он не будет разрешать авторизировать пользователей, которые исчерпали свой лимит.

Предположим, что пользователь `denis` оплатил 3 часа ($3 \cdot 3600 = 10800$ секунд) работы. Информацию об этом факте нужно добавить в таблицу `radcheck`:

```
> INSERT INTO radcheck VALUES('', 'denis', 'Max-All-Session', ':=', '10800');
```



Примечание

Если пользователь уже когда-то ранее оплачивал услуги, то есть запись `Max-All-Session` уже есть в таблице, нужно использовать оператор `UPDATE`, а не `INSERT`.

Кроме счетчика **sqlcounter**, есть счетчик **counter**, о нем вы сможете прочитать в документации по FreeRADIUS.

У счетчика **sqlcounter** есть один большой недостаток. Представим, что пользователь оплатил 3 часа работы. Он успешно проходит авторизацию и ... работает 5 часов вместо 3 положенных. К сожалению, счетчик не разрывает соединения по исчерпанию лимита, потому что запуск счетчика происходит при авторизации пользователя. Чтобы ваши пользователи не работали 24 часа в сутки, оплатив только 1 час, установите какую-нибудь систему управления трафика, основанную на SNMP (Simple Network Management Protocol). Подробное рассмотрение таких систем выходит за рамки этой книги.

Практически все готово, кроме ненастроенного маршрутизатора, который пылится в углу серверной комнаты. Маршрутизатор нужно прописать в файле `clients.conf` RADIUS-сервера, а на самом маршрутизаторе нужно указать следующие параметры

```
aaa new-model
ip radius source-interface Ethernet0
radius-server host 192.168.1.1 auth-port 1812 acct-port 1813
radius-server retransmit 10
radius-server timeout 3
radius-server deadtime 1
radius-server key our_secret
aaa authorization network radius
aaa authorization exec radius
aaa authorization login radius_auth local radius
aaa authentication ppp ppp_auth radius
aaa accounting update periodic 1
aaa accounting network default wait-start radius
aaa accounting connection default start-stop radius
```

28.2.5. Если что-то не работает

Рекомендую включить протоколирование ошибок в файле конфигурации `radiusd.conf` — потом будет проще отлаживать ошибки. Так как система FreeRADIUS довольно сложна, разработчики предусмотрели возможность ее тестирования — для этого существует утилита **radtest**. Формат ее вызова:

```
radtest user_name password radius_server ras_port secret
```

Можно также попытаться запустить сервис **radiusd** с опцией **-x**:

```
radiusd -x
```

В результате на консоль будут выведены отладочные сообщения.

Настройка обратного звонка (callback)

29.1. Что такое callback?

Первоначально обратный звонок был предназначен для снижения стоимости международных телефонных переговоров. Стоимость разговора определяется так: отсчет начинается с момента, когда вызываемый абонент поднял трубку или после пятого гудка, если абонент не отвечает, интервал тарификации — 1 минута, то есть каждая неполная минута будет оплачиваться как полная. Стоимость самого разговора зависит от страны, из которой мы звоним. Например, в Украине 1 минута связи с США вам обойдется в 2,5...3,3 доллара США в зависимости от типа линии, которую вы используете: обыкновенную или Utel. Звонок из США в Украину вам обойдется 1...2 доллара. На этом и основана идея **callback** (callback — обратный звонок).

Рассмотрим небольшой пример: звонок из Украины в США:

1. Украинский абонент набирает выделенный ему номер в США и после первого вызова кладет трубку. Этим звонком он активирует специальное оборудование системы **callback**. Естественно, этот звонок не оплачивается, потому что оплата начинается с момента поднятия трубки вызываемым абонентом или после пятого вызова. Украинский абонент должен быть подключен к линии с тональным набором.
2. Через 5...20 секунд система **callback** перезванивает абоненту Украины и приглашает его набрать номер, по которому он хочет позвонить. Система **callback** набирает этот номер. При этом соединение устанавливается из США, а не из Украины, что в конечном итоге ведет к снижению стоимости всего звонка. При этом интервал тарификации не одна минута, как при звонке из Украины, а всего 6 секунд.

В случае с доступом в какую-нибудь сеть система **callback** работает почти аналогично. Только сейчас нашей целью является не снижение сто-

имости (хотя это тоже не исключено), а повышение безопасности сети и дополнительная ее защита от несанкционированного доступа.

Работает **callback** примерно так: клиент сети, как обычно, устанавливает соединение с сервером и проходит аутентификацию. Если аутентификация прошла успешно, сервер обрывает соединения (кладет трубку). Естественно, если аутентификация не прошла, сервер кладет трубку и не предпринимает никаких дальнейших действий. Кроме логина и пароля, клиент также передает некоторое «волшебное» слово.

Если сервер получил это слово, то через определенное время (обычно 25...30 секунд) сервер обратного звонка перезвонит по заранее запрограммированному номеру клиента и установит соединение. После этого можно работать в сети как обычно. Как видите, если раньше для доступа в сеть достаточно было знать имя пользователя и пароль, то сейчас нужно, чтобы компьютер, который пытается войти в сеть, был подключен к телефонной линии с **заранее запрограммированным номером**.

Например, ваш сосед купил пакет неограниченного доступа к Интернет, а вы каким-то образом узнали его имя пользователя и пароль. При обычном соединении (не **callback**) вы можете работать в Интернете так же, как и сосед, но, естественно, в разное время. При использовании технологии **callback** подобное уже не пройдет, потому что система **callback** перезвонит не к вам домой, а к вашему соседу. Таким образом, система **callback** является дополнительным барьером для нежелательных гостей нашей сети.

В этой главе будут рассмотрены два варианта настройки системы **callback**. Второй вариант более простой и удобный для клиентов: для установления соединения не нужно никаких скриптов, но этот способ почему-то не всегда корректно работает. Первый требует специальной настройки клиентов, но работает в большинстве случаев. Я рекомендую вам поступить следующим образом: ознакомиться с обоими способами, а начать настройку сначала со второго способа и, если он у вас не будет работать, перейти к настройке первого.

29.2. Настройка сервера

29.2.1. Способ 1 — сложный, но надежный

Приступим к настройке по первому способу. Как я уже отмечал, этот способ предполагает дополнительную настройку клиентов, что не очень удобно. Например, если вы являетесь администратором компании-провайдера Интернета, и у вас несколько сотен пользователей, настраивать все машины, даже если эта работа оплачивается, не очень хочется. Можно, конечно, распечатать подробные инструкции для клиентов, но ведь существуют и такие пользователи, которые и обычный доступ настроить не в состоянии.

Я предполагаю, что программы **mgetty** и **pppd** у вас уже настроены и сам сервер удаленного доступа нормально функционирует. Напомню, что установка и настройка программы **mgetty** была подробно рассмотрена в предыдущей гл. 28.

При сборке программы **mgetty** нужно включить опцию автоматического распознавания **pap-авторизации**. Если вы этого не сделали, сейчас самое время это сделать. Если вы установили **mgetty** из пакетов RPM, то можете пропустить инструкции по включению этой функции. Хотя если у вас все-таки функция **DAUTO_PPP** работать не будет, даже при установке из пакетов RPM, вам придется пересобрать программу самостоятельно. Исходные тексты программы **mgetty** доступны по адресу <http://alpha.greenie.net/mgetty/>. После распаковки копируем файл `policy.h-Dist` в файл `policy.h`. В этом файле нужно сделать определенные изменения, которые подходят для вашей системы. В большинстве случаев нужно изменить расположение каталогов или же вообще ничего не изменять.

Теперь нужно включить автоматическое распознавание **pap-авторизации**. Для этого в файле `Makefile` найдите строку

```
CFLAGS=-O2 -Wall -pipe
```

и измените ее на

```
CFLAGS=-O2 -Wall -pipe -DOFIDO -DOAUTO_PPP
```

В этом же файле можно изменить установочные каталоги, но я не рекомендую этого делать. Оставьте все как есть: **SBINDIR=/sbin**, **BINDIR=/bin**

После этого выполните две команды, которые откомпилируют и установят **mgetty**:

```
make all
make install
```

Теперь перейдите в каталог **callback** каталога, который содержит исходные тексты **mgetty**. Скопируйте программу **callback** в каталог `/usr/sbin`, а файл `callback.config` — в каталог `/etc/mgetty+sendfax`. Отредактируем файл `/etc/mgetty+sendfax/login.conf`. Раньше (при условии, что вы настраивали сервер удаленного доступа) он у вас содержал примерно такую строку:

```
/AutoPPP/- a_ppp /etc/ppp/ppplogin
```

Сейчас эту строку нужно изменить на следующую:

```
/AutoPPP/ - a_ppp /usr/sbin/pppd noauth -chap +pap -detach
```

При обнаружении входящего звонка **mgetty** передаст управление демону **pppd**.

Добавим в файл `mgetty.config` описание порта, к которому подключен модем (см. листинг 29.1)

Листинг 29.1. Файл `mgetty.config`

```
port ttyS0
# режим для приема только данных (отключает факсимильные возможности)
dataonly y
# Строка инициализации модема
init-chat "" AT OK # Здесь можно задать строку инициализации модема
# Реинициализация модема
force-init-chat "" AT OK
```

Теперь нужно отредактировать файл `/etc/ppp/options.ttyS0` (см. листинг 29.2). В этом файле вы укажете параметры порта `ttyS0`.

Листинг 29.2. Файл `/etc/ppp/options.ttyS0`

```
# Максимальная скорость передачи данных
38400
# Аппаратный контроль передачи
crtsets
# Блокировка устройства ttyS0
lock
# Режим коммутируемой линии
modem
# IP-адреса сервера и клиента соответственно
192.168.1.1:192.168.1.207
# IP-адрес сервера DNS. Этот параметр обязателен для Windows-клиентов
ms-dns 192.168.1.1
# Интервал отправления LCP-пакетов
lcp-echo-interval 20
# Количество не принятых LCP-пакетов
lcp-echo-failure 6
# Размеры пакетов
mtu 576
mru 576
```

Интервал отправления LCP-пакетов и количество не принятых LCP-пакетов нужны серверу для определения функционирования клиента, то есть с помощью этих параметров **pppd** определяет, «жив» или нет клиент. Если на протяжении последних 120 секунд (20*6) клиент не прислал подтверждения, то сервер разорвет связь.

Запустите **mgetty** как обычно — через файл `/etc/inittab`:

```
S0:35:respawn:/sbin/mgetty -D -n 1 /dev/ttyS0 38400
```

Напомним, что **S0** — это просто идентификатор и вы можете использовать любое другое значение. 3,5 — это уровни запуска. Параметр **-D** программы **mgetty** указывает ей отключить все факсимильные возможности модема, а параметр **-n 1** заставляет **mgetty** поднимать трубку после первого звонка.

Теперь нужно запустить (перезапустить) **mgetty**. Для этого выполните команду **init q**. Также можно воспользоваться командой **killall -1 init**.

После этого на вашем модеме должна загореться лампочка DTR. Если этого не произошло, смотрите файл `/var/log/messages` — вы что-то сделали неправильно.

Помните, в самом начале я упомянул некоторое «волшебное» слово. Допустим, что этим «волшебным» словом будет слово «пожалуйста» (please). Вот его-то и нужно записать в файл `login.config`:

```
please -- /usr/sbin/callback -s 38400
```

Я позволил себе немного пошутить относительно выбора этого самого слова. Разницы, конечно, нет никакой — вы можете в качестве этого слова установить все, что вам угодно — даже свое имя. Обычно же используются слова «callback» или «cbuser» (callback user). После этого снова следует перезапустить **mgetty**.

Вот, собственно, в чем заключается первый способ настройки. Сейчас рассмотрим второй способ, а в пункте «Настройка клиентов» будем настраивать Windows-клиенты.

29.2.2. Способ 2 — простой, но не очень надежный

Второй способ, как я уже говорил, обладает неоспоримыми преимуществами. Во-первых, вам не нужно будет писать никакие сценарии для Windows-клиентов. Во-вторых, вы сможете самостоятельно определить, какие пользователи будут использовать функцию **callback**, а какие — нет.

Но и этот способ имеет свои недостатки. Хорошо, что существенный недостаток только один — невозможно авторизация, основанная на сценариях, как в первом способе, а иногда она бывает очень даже полезной. И еще один небольшой недостаток: нужно установить дополнительный патч к демону **pppd**, а так как патчи не всегда успевают за версиями **pppd**, то приходится использовать более старую версию **pppd**.

Сначала нужно выкачать патч к **pppd**, который реализует поддержку **callback**. Он доступен по адресу: <http://www.pbko.sk/~bobovsky/archiv/pppd-cbcpS-callback/ine-contrib/ppp-2.x.n.CBCP.patch>. Числа **x** и **n** — это номера версии демона **pppd**. Выкачивайте самую последнюю версию. Если у вас версии **pppd** старше, чем версия патча, вам придется установить более старую версию **pppd**. Различные версии **pppd** доступны по адресу <ftp://ftp.linuxcare.com.au/pub/ppp/>

Для обновления **pppd** (установки патча) используйте команду:

```
patch -p1 < ppp-2.3.10.CBSP.patch
```

Данная команда обновляет исходные тексты **pppd** (предварительно их нужно выкачать и установить). Эта же команда создает файлы:

```
/etc/ppp/callback-users  
/etc/ppp/callback-client  
/etc/ppp/callback-server
```

В первом из них нужно будет прописать всех пользователей, которым будет доступна функция обратного звонка. Второй нужен для работы у Linux-клиента функции **callback**. А третий управляет сервером обратного звонка.

Затем перейдите в каталог с исходными текстами **pppd** и введите три команды:

```
./configure  
make  
make install
```

После установки **pppd** нужно настроить **mgetty**. Напомню, что программа **mgetty** должна быть собрана с поддержкой функции **-DAUTO_PPP** (автоматическая rpp-авторизация).

Далее отредактируйте файл `/etc/mgetty+sendfax/login.conf`. Он должен содержать одну строку:

```
/AutoPPP/ - a_ppp /usr/sbin/pppd auth -chap +pap login callback-server
```

После этого пропишите своих пользователей в файле `callback-users` (см. листинг 29.3).

Листинг 29.3. Файл `/etc/ppp/callback-users`

```
# User list for callback  
# Username option  
# option - no callback  
# option * or empty user defined  
# option other admin defined: this number  
# in username * and ? wildcars valid, callback uses the best fit  
# Examples:  
# zoty 67435 # user zoty admin defined, number 67453  
# gates - # gates not called back may *  
cbuser *  
user 320779  
* -
```

Первый пользователь — `cbuser`. Согласно опции `*` — это пустое определение пользователя — для тестирования. Второй пользователь `user` —

это реальное определение пользователя, телефон для обратного звонка — 320779. Все остальные пользователи не будут использовать функцию `callback` — опция «-».

С помощью команды `chmod` сделайте сценарии `callback-server` и `callback-client` исполнимыми. После этого необходимо немного отредатировать скрипт `callback-server` (см. листинг 29.4.).

Листинг 29.4. Файл `/etc/ppp/callback-users`

```
#!/bin/sh
# Script callback-server
# Script parameters: delay time in seconds, callback number
DELAY="$1"
NUMBER="$2"
/usr/sbin/chat -v -t 2 "" ATN0
sleep $DELAY
/usr/sbin/chat -v "" AT OK ATS39=5DT$NUMBER CONNECT
```

Данная конфигурация уже должна работать, но иногда модем не успевает инициализироваться, поэтому после команды `sleep $DELAY` следует добавить еще одну команду `sleep`, например, `sleep 25`. Обратите внимание: используется тональный набор (АТ-команда **DT**). Напомню, что для импульсного набора используется команда **DP** (ATDP).

Вот и все, осталось только проверить корректность работы сервера.

29.3. Настройка клиентов

29.3.1. Способ 1

Создайте новое соединение и приступите к его конфигурации. При этом на вкладке «Тип сервера» выключите все параметры, кроме программного сжатия данных. Единственный допустимый сетевой протокол — TCP/IP (см. рис. 29.1)

Затем создайте в любом текстовом редакторе, например, в **Блокноте**, сценарий для обратного звонка (см. листинг 29.5).

Листинг 29.5. Сценарий для `callback`

```
proc main
delay 1
# это ваше "волшебное" слово
transmit "please^M"
# Ожидание запроса номера телефона
waitfor "phone"
```

```
# Передача номера
transmit "123456^M"
# Ожидание вызова
waitfor "RING"
# Ожидание соединения модемов
wainfor "CONNECT"
endproc
```

Сохраните свое творение как файл `callback.scp`. В операционной системе Windows NT данный файл нужно записать в каталог `\WINNT\SYSTEM32\RAS`.

После этого перейдите на вкладку **Сценарии** и выберите только что созданный сценарий (см. рис. 29.2).

Затем перейдите на вкладку **Общие** и нажмите на кнопку «Настройка». В появившемся окне перейдите на вкладку **Подключение** и нажмите на кнопку «Дополнительно». Далее в строке инициализации модема необходимо ввести `AT&C1S0=1` (см. рис. 29.3). Команда **&C1** устанавливает сигнал CD — без него этот способ работать не будет. Команда **S0** устанавливает количество звонков, после которых модем клиента будет снимать трубку (1 звонок). Для модема Motorola Premier 33.6 установите такую строку инициализации:

```
AT&F&C0S0=1Q0V1&D3\V4
```

В операционной системе Windows NT обратный вызов настраивается несколько иначе. С этой целью откройте окно запуска удаленного доступа:

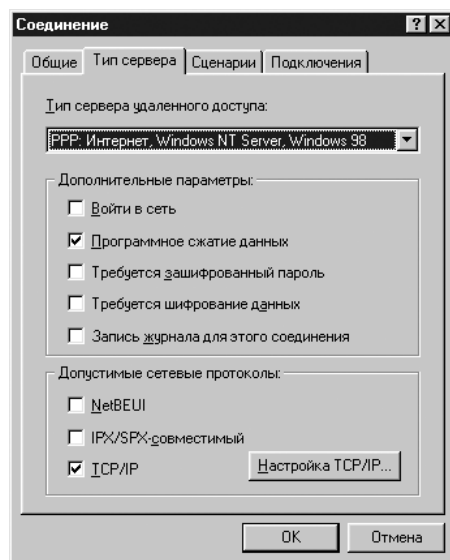


Рис. 29.1. Свойства соединения

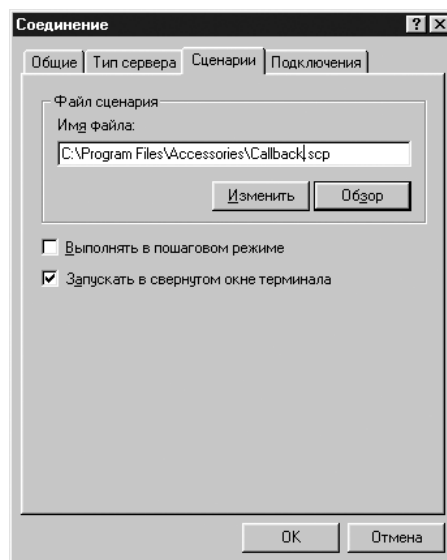


Рис. 29.2. Сценарий `callback.scp`

Пуск → Программы → Стандартные → Удаленный доступ (Start → Programs → Accessories → Remote access). Нажмите на кнопку «Другое» («More»). Выберите пункт меню «Параметры пользователя» («Users preferences») и перейдите на вкладку **Ответный вызов** («Callback»). Отметьте пункт «Да, требуется ответный вызов по указанным номерам». Номер телефонной линии, на которой установлен модем, можно изменить, нажав на кнопку «Изменить».

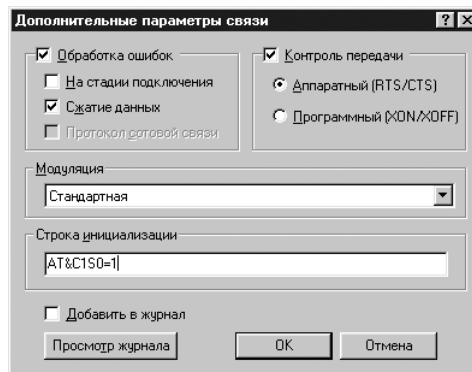


Рис. 29.3. Дополнительные параметры связи

29.3.2. Способ 2

Как я уже говорил, для второго способа не нужно создавать никаких сценариев для Windows-клиентов. И, как правило, никаких проблем с настройкой Windows здесь не возникает — нужно просто использовать обыкновенное соединение. Базовая настройка соединения производится так же, как и в первом случае (см. рис. 17.2).

Однако при использовании второго способа могут возникнуть проблемы с настройкой Linux-клиентов. На Linux-клиентах должна быть установлена та же версия **pppd**, что и на сервере. И так же, как и на сервере, ее необходимо пропатчить. После обновления демона **pppd** нужно настроить файл `/etc/ppp/callback-client` (см. листинг 29.6).

Листинг 29.6. Файл `/etc/ppp/callback-client`

```
#!/bin/sh
# Script callback-client
# Script parameters: delay time in seconds
DELAY="$1"
# Кладем трубку
/usr/sbin/chat -v -t 2 "" \d+++ \d\c OK ATH0 OK
# Вместо параметра $DELAY установите значение,
# которое подходит для вашего модема
# Подойдет delay 25 или даже delay 30
sleep $DELAY
# Ожидание callback
/usr/sbin/chat -v "" ATZ OK "" RING ATA CONNECT
```

В файле `ppp-on` нужно вызывать демон **pppd**, что можно сделать следующим способом:

```
/usr/sbin/pppd auth -chap +pap login callback
```

Linux-сервер для игрового зала

30.1. Достоинства и недостатки использования Linux в игровых залах

В этой главе будет рассмотрена настройка Linux как рабочей станции для игрового зала. У вас может возникнуть вопрос: почему именно как рабочей станции? Ответ очень прост: любую Linux-систему довольно легко превратить из рабочей станции в сервер, причем без потери надежности и производительности, чего нельзя сказать о Windows.

Итак, допустим, что у вас есть небольшой игровой зал, скажем, на 20...30 компьютеров и вам нужно по тем или иным причинам перейти на платформу Linux. Как я уже отмечал, любую из Linux-машин можно настроить как сервер и при этом можно использовать ее как рабочую станцию, то есть при этом не теряется ни одно пользовательское место при организации сервера.

Сейчас мы разберемся во всех достоинствах и недостатках (к сожалению, таковые имеются) такого преобразования. Достоинства и недостатки я буду приводить одновременно: сначала положительный момент, а затем — обратную сторону медали.

Достоинство. Самым большим достоинством является, на мой взгляд, существенная экономия денег, что немаловажно при открытии нового зала, когда первым делом нужно окупить средства, вложенные в его организацию.

Коробочные версии Windows XP Home Edition стоят около 160 долларов США. При открытии зала с парком в 30 машин общая стоимость боксовых версий обойдется вам примерно в \$ 4800. При покупке OEM-версий стоимость Windows составит около \$ 2400. При всем этом вы получите одноранговую сеть, состоящую из 30 компьютеров под управлением Windows. Если же вам нужно организовать сервер для доступа к Интернет, то за него придется выложить еще около \$ 1000. Итого \$ 5800. И

это только программное обеспечение — математика, а ведь еще нужно купить железо, дополнительную аппаратуру, мебель и т.д. Если нормальный компьютер для игрового зала можно купить за \$ 350...450, то зачем же увеличивать его стоимость даже на 80 долларов при использовании OEM-версии?

В случае с Linux вам достаточно купить один дистрибутив стоимостью \$5...10 долларов (при этом вы платите только за носители информации, то есть за компакт-диски, входящие в состав дистрибутива). Потом вы можете установить этот дистрибутив на неограниченное число компьютеров.

Недостаток. Несмотря на довольно приличную сумму сэкономленных денег, возрастут ваши ежемесячные расходы. Дело в том, что Linux-залу нужен квалифицированный системный администратор, хотя бы на первых порах — пока все не заработает так, как нужно. В этом случае услуги студента-первокурсника, пусть даже отлично знающего Windows, не будут соответствовать вашим запросам.

Установить Linux сможет каждый: современные программы установки Linux все сделают за вас. А вот настроить систему такой «специалист» вряд ли сможет, а поэтому вам понадобится специалист, хорошо знающий Linux. Следовательно, и зарплата у него должна быть как минимум в два-три раза больше, чем у администратора, обслуживающего одноранговую Windows-сеть. Кроме этого, понадобится определенное время на настройку всех компьютеров, так как настройка Linux занимает больше времени, чем Windows, а особенно настройка игровых приложений под Linux. Подробнее о переходе на Linux вы можете прочитать в моих статьях «Переходим на Linux» и «Строим бесплатный Интернет-сервер», которые вы найдете на сайте <http://dkws.narod.ru>.

Достоинство. Если вы имеете хотя бы небольшой опыт работы с Linux, вы уже должны были для себя отметить надежность работы этой операционной системы. А это значит, что вам или вашему администратору не нужно по 5...10 раз в день перезагружать машину из-за того, что «программа выполнила недопустимую операцию». Большинство современных игр являются сетевыми или же обладают поддержкой сети. Операционная система Linux работает с сетью гораздо быстрее, чем любая система семейства Microsoft.

Недостаток. Да, сетевые игры под управлением Linux работают быстрее. Но это относится только к Linux-играм. А на платформу Linux портировано не такое уж и большое количество игр. Самые популярные игры продолжают существовать только в Windows-варианте, поэтому запускать такие игры вам придется из-под эмулятора Windows, что сказывается на работе игры.

Во-первых, игры в родной Windows-среде работают стабильнее. Во-вторых, при работе из-под эмулятора игры основательно «притормаживают».

В-третьих, под управлением эмулятора работают далеко не все игры, хотя самые популярные все же работают. Конечно, все это в какой-то мере компенсируется более быстрой работой сети, но иногда даже не хочется играть, когда тебя «убивают» из-за того, что эмулятор не успел вовремя обновить экран.

Но в любом случае, игры под Linux работают, причем некоторые даже показывают довольно неплохие показатели, например, производительность Counter Strike под управлением эмулятора составили 83-88 fps, а под Windows 98 — 92-95 fps (использовалось разрешение 800x600 и драйвер видеокарты для Linux от компании nVidia). Конфигурация компьютера: AMD Athlon 700 Mhz/256MB/40GB Quantum 7200rpm/32MB/RivaTNT2 Pro.

Достоинство. При использовании Linux можно покупать не полноценные компьютеры, а только X-терминалы. В качестве X-терминала может выступать обыкновенный PC-компьютер без жесткого диска. Все программы, в том числе X-сервер и игры, будут выполняться на сервере, а пользователь увидит лишь результат выполнения программы. Естественно, в качестве сервера нужно купить довольно мощный компьютер. В начале книги, когда обсуждалась установка Linux, я писал, что при работе с Linux более критичен объем ОЗУ, чем частота процессора. В случае с сервером терминалов частота играет тоже довольно большую роль, потому что сервер должен будет обслуживать два-три десятка клиентов. При большом количестве клиентов целесообразно будет установить несколько серверов, скажем, один сервер на каждые 25 компьютеров. При этом предпочтительнее использовать двухпроцессорные конфигурации для сервера. Настройка X-терминалов рассматривалась в гл. 20 этой книги.

30.2. Использование эмулятора WineX для запуска Windows-приложений под Linux

30.2.1. Эмуляторы Wine и WineX

Стандартный эмулятор **wine** входит в состав практически любого дистрибутива, но он не обеспечивает должного уровня эмуляции операционной системы Windows. Для нормальной работы игр для Windows вам потребуется эмулятор **wineX** (и его следующие версии — **wineX2**, **wineX3**). Не путайте эмулятор **wine** с эмулятором **wineX**! Эмулятор **wineX** — это отдельная разработка и, к сожалению, этот эмулятор не является бесплатным — за него нужно платить.

Купить данный эмулятор можно на сайте <http://www.transgaming.com>. При покупке **wineX** у вас появится возможность загрузить уже скомпилированную версию эмулятора в виде пакета **rpm**. На этом же сайте можно бесплатно загрузить исходный текст эмулятора, но вы потратите много

времени на то, чтобы привести исходный код к пригодному для компиляции виду.

Устанавливать эмуляторы нужно в такой последовательности: **wine**, **wineX**, **wineX2**, **wineX3**. Напомню, что эмулятор **wine**, скорее всего, уже будет установлен у вас.

30.2.2. Установка Windows-игр (приложений) под Linux

Эмулятор **wine** гарантированно поддерживает следующие игры:

1. Counter Strike
2. StarCraft
3. Fallout
4. Fallout 2
5. Gunman
6. Quake 2
7. Quake 3
8. Soldier of Fortune
9. Unreal Tournament
10. Red Alert (все версии)
11. Diablo 2
12. Cesaer
13. Return to Castle Wolfenstein
14. Star track
15. Kingpin
16. Nox
17. Jadded Alliance
18. 4x4 Evolution
19. American McGee Alice
20. Daikatana
21. Heroes of Might and Magic III
22. Delta Force 1,2

Возможно, у вас будут работать и другие игры.

Перед установкой игры удалите все файлы с расширением **reg** в подкаталоге **.wine** домашнего каталога пользователя **root**:

```
# rm -rf /root/.wine/*.reg
```

Запустите сервер X, если он еще не запущен командой:

```
startx
```

Если сервер X загружен, но вы работаете в консоли, перейдите в графический режим и запустите графический эмулятор терминала, например, `xterm`. Для установки новой игры выполните команду:

```
# wine install_program
```

Предположим, что программа установки игры называется `setup.exe` и находится в корневом каталоге компакт диска. Для установки такой игры нужно ввести команду:

```
wine /mnt/cdrom/setup.exe
```

Игра будет установлена в каталог `/usr/local/wine-c/games/<название_игры>` или же в каталог `/usr/share/wine-c/games/<название_игры>`. Узнать, в какой из этих двух каталогов была установлена игра, вы можете, просмотрев файл `/root/.wine/.config`. В секции **Drive C** определяются настройки для диска C:

```
[Drive C]
«Path» = «/usr/share/wine-c»
«Type» = «hd»
«Label» = «MS-DOS»
«Filesystem» = «win95»
```

Пользовательские настройки эмулятора находятся в файле `config`, который находится в каталоге `$HOME/.wine`. Глобальные настройки эмулятора вы можете изменить в файле `/etc/wine.reg`.

30.2.3. Запуск Windows-игр под Linux

После установки игры перейдите в каталог, в который была установлена игра, то есть в каталог `/usr/share/wine-c/games/<название_игры>`. Попробуйте запустить ее, поочередно используя команды:

```
wine game.exe
winex game.exe
winex2 game.exe
winex3 game.exe
```

Естественно, вместо параметра **game.exe** нужно подставить реальное имя исполняемого файла игры. Данные команды нужно вводить в терминале X, например, **kterm**, если вы используете KDE. Если игра не запустилась, ее следует удалить. Для этого просто удалите каталог `/usr/share/wine-c/games/<название_игры>`. Если игра запустилась, вы должны увидеть окно эмулятора **wine** (см. рис. 30.1).

Желательно сразу же открыть окно настроек программы и поэкспериментировать с настройками видеорежимов. Например, Unreal Tournament у меня намного быстрее работал при использовании программного рендеринга (Software Rendering), чем при использовании драйвера Direct3D.



Рис. 30.1. Окно эмулятора wine

Теперь приступим к настройке запуска игры. Скопируйте каталог /root/.wine в каталог /root/.wine_<название игры>. Создайте файл /root/<название игры>_run:

```
touch /root/<название игры>_run
```

Содержимое этого файла зависит от эмулятора, с помощью которого запустилась игра (**wine**, **wineX**, **wineX2**).

Для **wine** содержимое файла будет таким:

```
export WINEPREFIX=$HOME/.wine_<название игры>
cd «/usr/local/wine-c/games/<название игры>»
wine <исполняемый файл игры> <параметры>
```

Для **wineX**:

```
export LD_LIBRARY_PATH=/usr/local/wineX/lib:$LD_LIBRARY_PATH
export PATH=/usr/local/wineX/bin:$PATH
export WINEPREFIX=$HOME/.wine_<название игры>
cd «/usr/local/wine-c/games/<название игры>»
wineX <исполняемый файл игры> <параметры>
```

Для **wineX2**:

```
export LD_LIBRARY_PATH=/usr/local/wineX2/lib:$LD_LIBRARY_PATH
export PATH=/usr/local/wineX2/bin:$PATH
export WINEPREFIX=$HOME/.wine_<название игры>
cd «/usr/local/wine-c/games/<название игры>»
wineX2 <исполняемый файл игры> <параметры>
Введите команду для изменения прав доступа:
chmod u+x < название игры >_run
```

Теперь для запуска игры можно использовать команду **/root/<название игры>_run**.

После установки всех игр удалите библиотеки Microsoft, которые будут установлены в каталог **/usr/local/wine-c/system**. Иногда эти библиотеки устанавливаются и в другие каталоги, поэтому внимательно изучите содержимое каталога **/usr/local/wine-c** и удалите лишние файлы.

Выполните команду **chmod -R o+w /usr/local/wine-c**. Эта команда установит права доступа к каталогу **/usr/local/wine-c**, которые позволят записать в играх и сохранение конфигураций пользователям.

Для включения полноэкранного режима установите значение переменной файла **/root/.wine/config** **Managed**, равное **N**, а также закомментируйте переменную **Desktop**:

```
; Allow the window manager to manage created windows
«Managed» = «N»
; Use a desktop window of 640x480 for Wine
; «Desktop» = «800x600»
```

После того, как все будет настроено, создайте пользователя **game**. Используя эту учетную запись, посетители игрового зала будут регистрироваться в системе. Скопируйте все файлы настроек в каталог **/home/game** и установите должным образом права доступа. Для этого можете использовать следующие команды:

```
cp /root/*_run /home/game
cd /home/game
chmod o+x *_start
cd /root/Desktop/* /home/game/Desktop
chown -R game:game /home/game/Desktop
mkdir /home/game/.kde/apps/share/WINE
cp -R /root/.kde/apps/share/WINE /home/game/.kde/apps/share/WINE
chown -R game:game /home/game/.kde/apps/share/WINE
```

Теперь пользователь **game** сможет запускать установленные вами игры. Как всегда, существует одна маленькая деталь, о которой постоянно забываешь: попробуйте объяснить посетителю, не знающему даже, как правильно завершить работу в Windows 98, что такое терминал **xterm** и что

для запуска игры **quake** нужно ввести команду **quake_run**. Вы правы, это будет довольно сложно, поэтому, чтобы не усложнять себе жизнь, каждый день отвечая на вопросы наподобие: «а как запустить этот самый xterm?» и чтобы не шокировать посетителей, создайте на рабочем столе ярлыки для всех файлов *_run.

Для этого щелкните правой кнопкой мыши на рабочем столе KDE и выберите команду **Создать→ Ссылка на приложение** (см. рис. 30.2).

В качестве рабочего стола по умолчанию я рекомендую использовать именно KDE, потому что этот рабочий стол максимально приближен к стандартному рабочему столу Windows и при работе с ним посетители будут задавать меньше вопросов.

После этого введите название игры и выберите для нее значок. Затем перейдите на вкладку **Выполнить** и выберите файл для запуска (см. рис. 30.3).

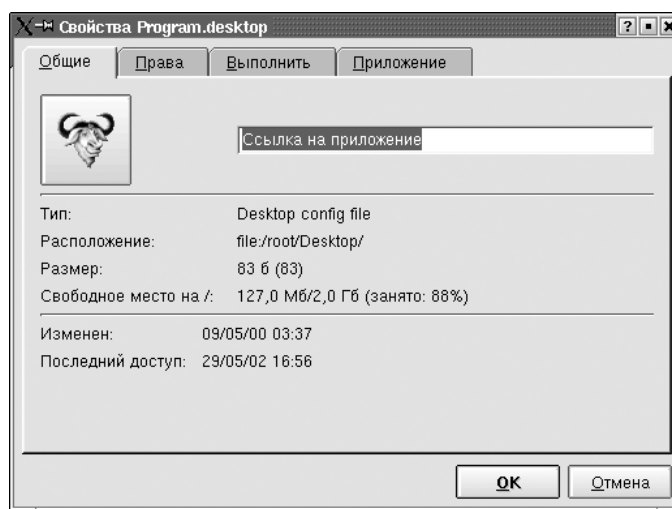


Рис. 30.2. Новая ссылка на приложение

Из рис. 30.3 видно, что при щелчке на этом ярлыке будет запущена игра **UNREAL** (файл **unreal_start**). Я рекомендую запускать данные файлы в терминале. Для этого можете включить режим **Запускать в терминале** и ввести параметры терминала, а можете просто ввести вместо команды **/home/game/unreal_start** команду **xterm -e /home/game/unreal_start**. Данная команда запустит терминал xterm, который использует небольшое количество системных ресурсов, а в этом терминале и будет запущен нужный вам файл запуска игры.

На вкладке **Права** вы можете установить права доступа к ярлыку (см. рис. 30.4). Обычно здесь ничего не нужно изменять.

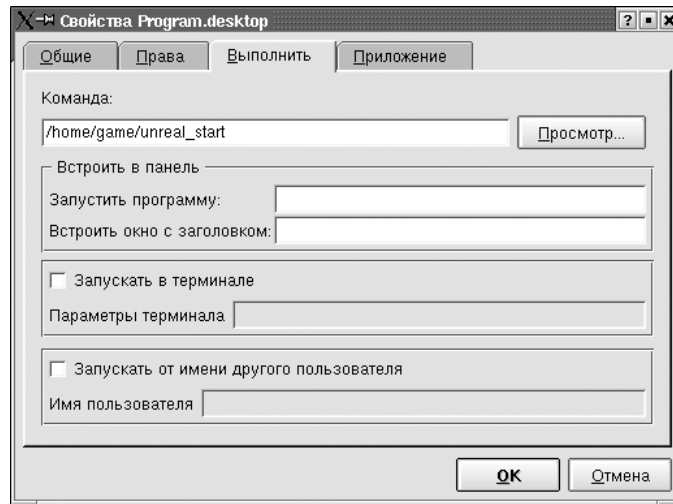


Рис. 30.3. Выбор игры

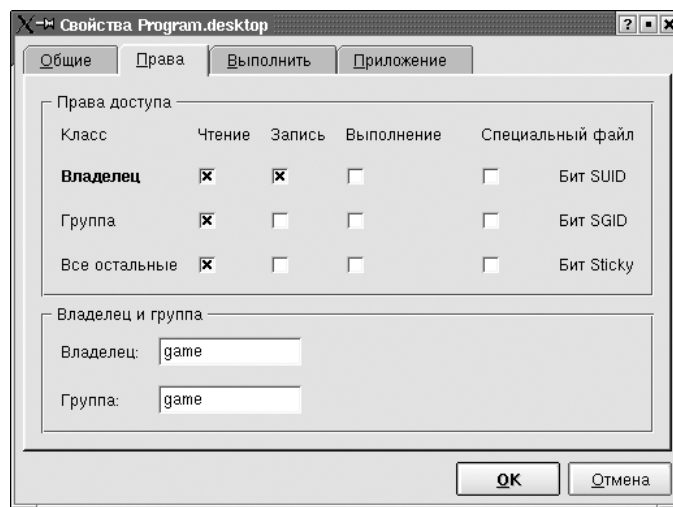


Рис. 30.4. Права доступа

Можно также создать ярлыки для других часто используемых программ: браузера **Mozilla**, клиента **licq**, пакета **Star (Open) Office**, проигрывателя **XMMS**.

30.3. Администрирование игрового зала

30.3.1. Введение

Вы уже справились с самой сложной задачей — настроили рабочее место посетителя. По сравнению с этой задачей администрирование игрового зала является второстепенным вопросом. Цель любого игрового зала — это получение прибыли, а последнее возможно лишь при условии, что:

1. Все игры будут работать, причем они должны работать быстро и без сбоев.
2. Сеть работает без сбоев.
3. Графический интерфейс пользователя интуитивно понятен.
4. Можно слушать MP3 и смотреть Mpeg4, а также проигрывать аудио компакт-диски.

Другими словами, клиенты будут посещать ваш зал, если в нем будет создана соответствующая обстановка. А каким образом вы администрируете ваш игровой зал, посетителей мало интересует. Например, если у посетителя вышло время, можно просто подойти и сказать ему об этом. Конечно, если такое позволяют размеры вашего игрового зала. В самом деле, не будете же вы идти через весь зал, чтобы сообщить посетителю номер 47, что ему уже нужно уходить или доплатить за дополнительное время? Можно автоматизировать этот процесс и автоматически отключить его от системы через определенное время.

В этой главе я сделал все возможное, чтобы описать запуск игр под Linux, и, я надеюсь, что игры в вашем игровом зале будут работать достаточно быстро. О сети позаботится сама операционная система, как вы уже знаете, реализация стека протоколов TCP/IP в операционной системе Linux намного эффективнее, чем в Windows.

Разработчики оконных сред KDE и Gnome позаботились о интуитивности пользовательского интерфейса, решив за нас этот вопрос. А об использовании средств мультимедиа говорить не нужно, так как это в Linux делается элементарно.

30.3.2. Доступ к Интернет в игровом компьютерном зале

Прежде всего, определимся нужен ли вашему игровому залу доступ к Интернет. Если нужен, то для каких целей. В большинстве случаев он нужен для того, чтобы посетители могли использовать так называемые Online-версии игр или подсоединяться к всемирным игровым серверам. Такую возможность предоставляют разработчики многих современных игр. Возможно, ваш зал — это не просто игровой зал, а еще и Интернет-кафе.

Итак, мы выяснили, что может быть три варианта:

1. Зал без доступа к Интернету.
2. Зал с доступом к Интернету для Online-игр.
3. Интернет-кафе.

В первом случае вы сразу можете перейти к следующему пункту — «Управление доступом». Сейчас же будет рассмотрена настройка сервера во втором и в третьем случаях.

Если вам нужно обеспечить только работу Online-игр, вам нужно будет установить и сконфигурировать такие службы:

1. **IPChains** или **IPTables** (в зависимости от версии ядра).
2. Прокси-сервер **Socks5**.

Настраивать бастион нужно в любом случае — он обеспечивает безопасность вашей внутренней сети. При настройке бастиона учитывайте особенности используемых вами игр. Например, выделенный сервер игры **Unreal Tournament** использует 7777 порт. На бастионе нужно будет разрешить порт 7777, если вы хотите, чтобы к вашему серверу могли подключиться извне, например, из другого игрового зала. Настройка бастионов уже обсуждалась в одноименной гл. 22 — «Бастионы».

Сервер **Socks5** нужно настроить только в том случае, если ваша игра требует реальный IP-адрес. Сервер **Socks5** нужен еще для организации рабочего места администратора, чтобы он на протяжении рабочего дня мог общаться со своими знакомыми по ICQ.

В третьем случае (Интернет-кафе) вам нужно настроить такие службы:

1. Бастион.
2. Прокси-сервер **SQUID**.
3. Сервер **DNS**.
4. Сервер **Socks5**.
5. Web-сервер.
6. Почтовый сервер.

Первые три службы вам нужно настроить обязательно, а все остальные — по вашему желанию. Как уже было отмечено, бастион нужен из соображений безопасности. Сервер **SQUID** нужен для кэширования Web-страниц клиентов, при этом совсем не обязательно устанавливать на сервере Web-сервер.

Сервер **DNS** также необходим для повышения производительности. Вы можете использовать сервер **DNS** вашего провайдера, однако, если вы настроите собственный сервер **DNS**, то:

1. Повысите скорость разрешения имен **DNS**.
2. Сэкономите на трафике.

Желательно настроить и почтовый сервер для повышения скорости отправки сообщений посетителей. Можно опять же так использовать или сервер провайдера или какой-нибудь бесплатный SMTP-сервер, например, smtp.mail.ru, но использование собственного сервера будет удобнее и дешевле.

Теперь, когда Интернет-сервисы уже настроены и у каждой рабочей станции есть доступ к Интернет, можно приступить к теории управления пользователями.

30.3.3. Управление пользователями

Вырабатываем политику управления пользователями

Сначала разберемся, что мы подразумеваем под управлением пользователями. Обычно все управление заключается в отслеживании времени работы посетителя и когда его время вышло, сообщении ему об этом. Естественно, если пользователей много, проследить за каждым — это довольно трудная задача. Даже если у вас будет журнал, в котором вы будете записывать время работы каждого пользователя, через пару дней вам основательно надоест каждые десять минут проверять, у какого посетителя вышло время. Например, если в вашем распоряжении 30 компьютеров, вам нужно будет каждые 10 минут просматривать все 30 записей.

Управлять пользователями можно по-разному. Можно по истечении определенного времени просто «отрубить» пользователя от системы. На что в ответ вы получите массу жалоб, и вряд ли ваш зал будет пользоваться популярностью при таком управлении. Вы, конечно, можете привести аргументы в свое оправдание: мол, он (посетитель) знает, что оплатил один час и должен «чувствовать» время.

Однако в нашем случае нужно учитывать тот факт, что у игрока отсутствует это самое «чувство времени», во время игры он не ощущает, прошло полчаса или пятьдесят минут. Поэтому, скорее всего, посетитель не успеет сохранить игру до того, как его отключат от системы. Можно предупредить посетителя о таких правилах, но при этом вы заставляете его быть в постоянном напряжении, постоянно поглядывая на часы. От такой игры никто не получит удовольствие.

Мы уже пришли к выводу, что теория «жесткого» управления нам не подходит, и сейчас рассмотрим более лояльный способ управления. Через определенное время, например, за пять минут до того, как у посетителя выйдет время, мы предупредим его об этом. За это время посетитель успеет сохранить игру и доплатить, если он захочет продолжить игру. Думаю, пять минут будет вполне достаточно, чтобы дойти к столу администратора.

Все вышеописанные функции выполняются специальным программным обеспечением для игровых залов. Для игровых залов, использующих операционную систему Windows, создана масса программ такого рода. К сожалению, мне не встречался нормальный пакет программ управления игровым залом для Linux. Можно было бы использовать **K12 Linux Terminal Server**, но этот программный комплект больше подходит для управления учебным классом, чем для управления игровым залом. В нем есть много ненужных функций, которые вы вряд ли будете использовать. Вам нужна программа, которая:

1. Предупредила бы посетителя, что через определенное время ему нужно освободить место.
2. Через определенное время «отрубила» бы его от системы.
3. С помощью которой вы могли бы отправить сообщение любому посетителю.

Как видите, для вас вполне достаточно трех этих функций. Аналогичное программное обеспечение ведет также протокол: кто, когда и сколько работал. Нам же эта функция не нужна, потому что протоколы ведет сама Linux (точнее, программы протоколирования). В любой момент вы можете посмотреть, кто и сколько работал. Например, узнать, когда регистрировался и сколько времени отработал в системе пользователь `den` можно с помощью команды (см. рис. 30.5):

```
last den
```

Аналогично если вы введете команду **last** без параметра, то увидите полный отчет о времени работы пользователей. Узнать время последней регистрации пользователя можно с помощью команды **lastlog** (см. рис. 30.6). Программы **last** и **lastlog** являются средствами просмотра файла `/var/log/lastlog`, который нельзя просмотреть «невооруженным глазом».

Пишем программу-launcher для автоматизированного управления пользователями

Вернемся к нашему программному обеспечению для управления посетителями. В силу невозможности найти какое-нибудь достойное уже созданное программное обеспечение я решил написать свою «программу» для управления игровым залом. Данное решение не претендует на первое место среди программ такого рода, но обладает всеми необходимыми функциями и достаточно простое в обращении. Обычно программы такого рода состоят из двух частей: модуль-клиент и модуль-сервер.

Модуль-клиент обычно установлен у администратора, и он может управлять множеством компьютеров локальной сети. Модуль-сервер запускается на компьютере посетителя и опрашивает некоторый порт. Как только модуль-сервер получил от администратора команду, он выполняет определенные действия, например, при получении команды **timeout** он отсоединяет пользователя от системы.

```
[root@localhost root]# last den
den      tty1          Tue Jun  4 16:35 - down      (00:05)
den      pts/1         Tue Jun  4 15:12 - down      (00:10)
den      :0           Tue Jun  4 15:12 - 15:13    (00:01)
den      :0           Tue Jun  4 15:01 - 15:11    (00:09)
den      :0           Tue Jun  4 14:57 - 14:58    (00:00)
den      pts/1      :0           Tue Jun  4 14:54 - 14:54    (00:00)
den      pts/0      :0           Tue Jun  4 14:49 - 14:57    (00:07)
den      pts/0      :0           Tue Jun  4 14:42 - 14:49    (00:07)
den      :0           Tue Jun  4 14:41 - 14:57    (00:16)
den      tty2          Mon Jun  3 14:42 - down      (00:00)
den      tty2          Mon Jun  3 14:40 - 14:41    (00:00)
den      tty2          Mon Jun  3 14:38 - 14:40    (00:02)

wtmp begins Tue May 28 15:12:02 2002
[root@localhost root]#
```

Рис. 30.5. Журнал регистрации

```
wtmp begins Tue May 28 15:12:02 2002
[root@localhost root]# lastlog
Username      Port      From      Latest
root          :0        -         Чтв Июн  6 11:51:58 +0300 2002
bin            -         -         **Never logged in**
daemon        -         -         **Never logged in**
adm           -         -         **Never logged in**
lp            -         -         **Never logged in**
sync          -         -         **Never logged in**
shutdown      -         -         **Never logged in**
halt          -         -         **Never logged in**
mail          -         -         **Never logged in**
news          -         -         **Never logged in**
uucp          -         -         **Never logged in**
operator      -         -         **Never logged in**
games         -         -         **Never logged in**
gopher        -         -         **Never logged in**
ftp           -         -         **Never logged in**
nobody        -         -         **Never logged in**
mailnull      -         -         **Never logged in**
rpm           -         -         **Never logged in**
xfs           -         -         **Never logged in**
ntp           -         -         **Never logged in**
rpc           -         -         **Never logged in**

[root@localhost root]#
```

Рис. 30.6. Время последней регистрации

В предлагаемом мною решении модуль-клиент, как и модуль-сервер, отсутствуют. Сейчас разберемся, почему. Мы настраиваем основной сервер так, чтобы к нему подключались все остальные компьютеры в сети — компьютеры посетителей. Поскольку пользователь уже зарегистрирован в нашей системе, для того, чтобы отключить его, достаточно просто локально «прибить» процесс этого пользователя. Под процессом следует понимать оконный менеджер данного пользователя.

Естественно, все компьютеры сети будут X-терминалами вашего сервера. Настройка X-терминала обсуждалась в гл. 20. При настройке руководствуйтесь такими правилами. Имя пользователя должно совпадать с именем рабочей станции. Например, если имя рабочей станции `game1`, то на этой станции должен быть зарегистрирован пользователь `game1`. На сервере должны быть зарегистрированы все пользователи: `game1`, `game2`, ..., `gameN`. Пароли установите по своему усмотрению, но пароли пользователей на сервере и на рабочих станциях тоже должны совпадать.

Все это необходимо для регистрации пользователя на сервере. Если настройка X-терминала показалась вам слишком сложной, сейчас рассмотрим более простой путь. В гл. 20 рассматривалась настройка «чистого» X-терминала, то есть загрузка X-терминала осуществлялась по сети, а на самом компьютере даже не был установлен жесткий диск. Сейчас же мы попытаемся настроить «условный» X-терминал. Почему условный? Операционная система будет устанавливаться на компьютеры посетителей как обычно, вместе с системой X Window. Затем в файле `/etc/initab` вы заменяете строку

```
X:123456:respawn:/usr/bin/X11/X
```

на строку

```
X:123456:respawn:/usr/bin/X11/X -query 192.168.0.1
```

Данная команда (`X -query 192.168.0.1`) обеспечивает загрузку системы X по умолчанию (уровень выполнения 5) и при этом будет использоваться сервер X с IP-адресом 192.168.0.1. Несложно догадаться, что компьютер с таким адресом — это и есть ваш сервер. Настройку сервера терминалов выполните так, как описано в гл. 20. При этом на сервере и клиенте желательно установить одну и ту же версию системы X Window.

Если на всех компьютерах установлено одно и то же оборудование, а в большинстве случаев это так, поступите таким образом: настройте систему X Window только на сервере, а затем обеспечьте доступ по NFS клиентам к файлам системы X Window. В этом случае на компьютере клиента вообще не нужно устанавливать систему X Window, а запускать ее непосредственно с сервера по сети, используя NFS. Настройка сетевой файловой системы (NFS) обсуждалась в гл. 8.

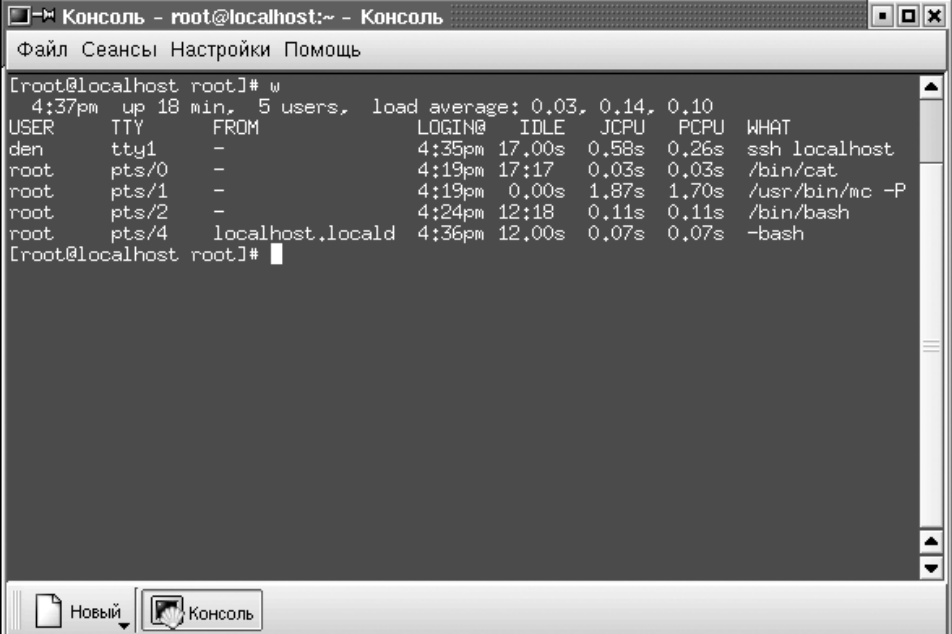
Я рекомендую использовать именно второй способ. Запуск игр тоже можно осуществлять по сети, предварительно расположив их в каталоге, доступном

по NFS. Естественно, для запуска и нормальной работы игр по сети нужна сеть, обеспечивающая скорость передачи данных 100 Мбит/с. Концентраторы (hub) в данной сети лучше заменить коммутаторами (switch).

Теперь перейдем к написанию самой программы. Данную программу мы напишем, используя «подручные» средства: стандартные программы Linux и командный язык интерпретатора **shell**. Во-первых, командный язык интерпретатора **bash** уже рассмотрен в этой книге. Во-вторых, если написать эту программу на C или Pascal, то читатель должен владеть данным языком программирования, что усложнит чтение книги.

Просмотреть всех зарегистрированных в системе пользователей можно с помощью команды **w** (см. рис. 30.7).

С помощью данной команды можно выяснить, сколько времени работает пользователь, использование процессора пользователем, какая программа выполняется в данный момент, а также общую загрузку системы (load average). Кроме другой полезной информации, команда **w** сообщает нам, с какой машины произошла регистрация пользователя в нашей системе. Будем рассматривать случай, когда имя пользователя будет совпадать с именем машины, что впоследствии значительно упростит вам администрирование залом.



```
[root@localhost root]# w
 4:37pm  up 18 min,  5 users,  load average: 0.03, 0.14, 0.10
USER    TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
den     tty1     -               4:35pm   17.00s  0.58s  0.26s  ssh localhost
root    pts/0    -               4:19pm   17:17   0.03s  0.03s  /bin/cat
root    pts/1    -               4:19pm   0.00s   1.87s  1.70s  /usr/bin/mc -P
root    pts/2    -               4:24pm   12:18   0.11s  0.11s  /bin/bash
root    pts/4    localhost.locald 4:36pm   12.00s  0.07s  0.07s  -bash
[root@localhost root]#
```

Рис. 30.7. Команда **w**



```
ps --user username
```

Теперь рассмотрим исходный текст этой программы (см. листинг 30.1).

```
#!/bin/bash
# Управление игровым залом — добавление нового клиента
# Распространяется по лицензии GPL
# (c) 2004 Denis Kolisnichenko, dhsilabs@mail.ru
# Шрифт для отображения сообщения
FONT="-cronyx-fixed-***-***-***-***-koï8-r"
# Размеры окна
GM="700x70"
# Сообщение
MSG="Ваше время вышло. В течение 5 минут вы можете оплатить до-
полнительное время"
if [ $# -lt 4 ]; then
```

Часть VI. Практические примеры полной настройки Linux-сервера

```
{ echo "Usage: newclient warntime time user num";
exit 1;
}
fi;
# Спим
sleep $1
# Отображаем предупреждение, поскольку время warntime вышло
xmessage -display server:$4 -fn $FONT -geometry $GM -bg black -fg
green $MSG
sleep $2
P='ps -user $3 | grep -i gnome-session | /bin/awk -F " " '{ print
$1 }''
echo $P
# Убиваем сессию пользователя
kill -9 $P
echo "Time of user $3 is out"
Запуск программы:
newclient 3540 3600 game1 1
newclient 55m 60m game2 2
newclient 55m 1h game3 3
newclient 23h 1d game4 4
```

При запуске программы нужно указать четыре параметра. Первый из них — это время, через которое будет отображено сообщение. Сообщение можно изменить по своему вкусу, отредактировав значение переменной MSG (так же, как и другие переменные). Следующий параметр — это время, по истечении которого пользователь будет «отрублен» от системы. Время можно указывать в секундах, в минутах (суффикс m), часах (суффикс h), днях (суффикс d).

Третий и четвертый параметры — это соответственно имя пользователя и номер дисплея. Номер дисплея будет совпадать с номером пользователя, если вы настроите систему согласно моим рекомендациям. Например, если имя пользователя game1, то номер дисплея — 1.

Обратите внимание на следующую строку программы:

```
xmessage -display server:$4 -fn $FONT -geometry $GM -bg black -
fg green $MSG
```

Данная строка обеспечивает отображение сообщения MSG на дисплее с номером \$4 компьютера server. X-терминал посетителя будет подключен как раз к дисплею с номером \$4. При этом посетитель увидит на экране примерно то, что показано на рис. 30.9.

Следующий аспект, на который вам нужно обратить внимание — это оконная среда пользователя. Если пользователь использует среду Gnome,

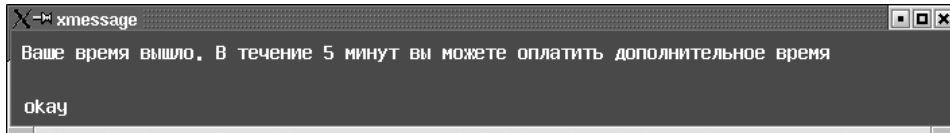


Рис. 30.9. Предупреждение об истечении времени

то в списке процессов пользователя будет процесс `gnome-session`. Если завершить этот процесс, пользователь будет отключен от системы. На этом и основывается данный метод работы программы. В листинге 21.2 подразумевается, что пользователь используется среду Gnome:

```
P='ps --user $3 | grep -i gnome-session | /bin/awk -F " " '{ print $1 }''
```

Если ваши посетители используют среду KDE, измените это строку на аналогичную ей:

```
P='ps --user $3 | grep -i kdeinit | /bin/awk -F " " '{ print $1 }''
```

Как проконтролировать, какую среду использует посетитель? Очень просто: при установке системы установите одну из сред: или KDE или Gnome. Можно также изменить исходный текст программы `newclient`, добавив соответствующую проверку, но зачем усложнять себе жизнь?

Еще раз рассмотрим запуск программы. Данную программу можно запускать в фоновом режиме, освободив консоль:

```
newclient 55m 60m game1 1 &
```

Эта команда будет выполняться в фоновом режиме. Как только выйдет время (1 час), на консоли вы увидите сообщение:

```
Time of user game1 is out
```

30.3.4. Ограничение доступа пользователей

Операционная система Linux обладает достаточно высокими средствами защиты информации, поэтому, используя стандартную конфигурацию (обыкновенный пользователь, а не суперпользователь), вы обеспечите высокий уровень безопасности. Другими словами, по поводу безопасности можете не волноваться: пользователь все равно не сделает ничего такого, что может повлечь за собой разрушение системы.

Единственное, что я могу порекомендовать, переместите файлы `/usr/bin/mc` и `/usr/bin/kcontrol-panel` в каталог пользователя `root`: посетителю незачем изучать аналог Norton Commander для Linux и тем более настраивать среду KDE.